PERSONAL COMPUTER WORLD

50p U.S.$2.00

THE JANUS INTERFACE

DRAWPIC

THE BYTE CONSERVER

RANDOM WRITINGS

FEBRUARY 1979

Europe's first magazine for personal computers for home and business use

A Mighty Micromite In Action

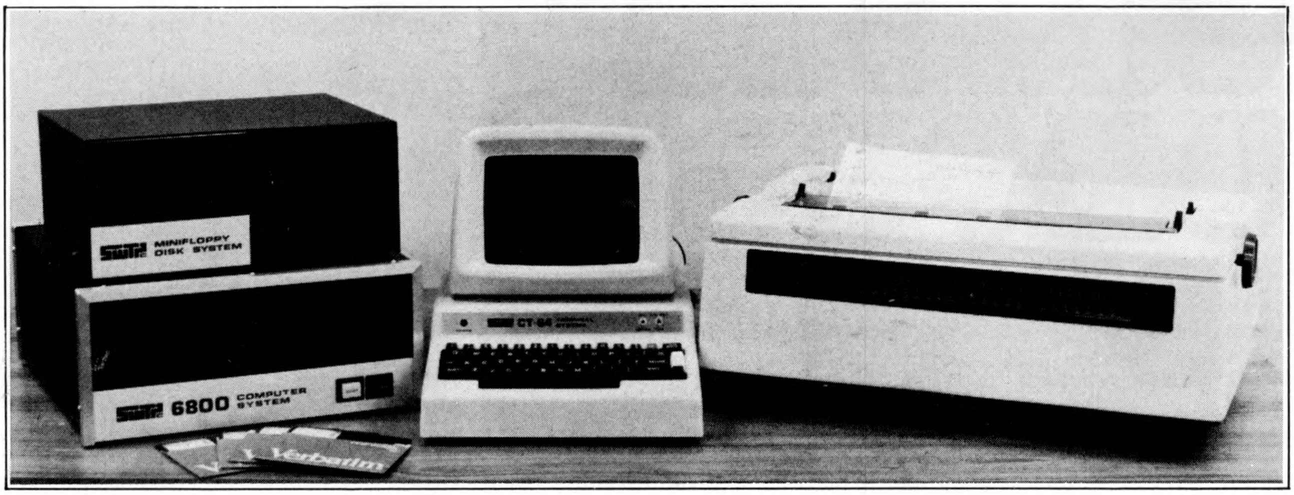
Setting Up A Local Group

The Circuit Inspector

The Attache

One of a growing band of small business machines

Anniversary Issue



He had never seen my computer and he was obviously impressed by the pile of perfectly typed overdue account letters it had just produced.

"How can you possibly afford a computer system in such a small company?" he asked, in that direct way suppliers have when they think that you may be overspending.

I had been anticipating the question. I had seen him glancing enviously at the Cash Flow Forecast, Sales analysis Report, and Back Order Schedule I had been referring to since he arrived.

He had realised that this was the first time ever that I had been able to put my finger on the facts which I need to schedule my next three months deliveries from him.

"I'll buy you lunch if you can get within £1,000 of the cost of the system," I said, generously, because it was his turn to pay today. *"And I'll tell you as much about it as you want to know."* I added.

"Well I can see it does the job one of those word processing machines does, and it's doing most of your accounting – but what does it actually consist of?"

Here was my chance to impress him with my very limited knowledge of the equipment itself.

"Well – here's the visual display terminal with the keyboard. As you can see it has upper and lower case characters and you use the keyboard like a typewriter. This box here is the computer itself which has 40K of RAM," I said quickly because that's all I know about it and I was hoping to avoid his next question. However, he butted in –

"What does that mean?"
"Er, well it's the amount of memory it's got."
"It couldn't be much in a box that size," he said.

"Well all I know is that it certainly seems to be enough to cope with any of the programs I use," I said defensively, *"and besides these disk drives hold over half a million characters of information which the computer can read whenever it needs them."*

"What's that in terms of names and addresses for instance?" he asked.

"Assuming 150 characters for each one it's about £3,800. And this is the printer which gives a typewriter quality letter or report."

"What else can it be used for?" he asked.
"Well this system is the top end of the range," I said proudly, *"but other cheaper models are used for everything from process control to medical interviewing, from playing games to student instruction, and from statistical analysis to travel booking."*

"You'll be telling me it can talk next," he said with a hint of sarcasm in his voice.

"Oh did I forget to mention that?"
"Oh no, you've told me enough already – I know it must be cheaper than I would expect because otherwise you couldn't have afforded it, without being rude, but even so it must have cost at least £10,000."

"Well you're right," I said tantalisingly, *"it is cheaper than you would expect. Even with the Speech unit it only cost me £5,673.24 including the Chancellor's 8%."*

"How come I always end up buying you lunch?" he said.

SWTPC Computer Systems

AVAILABLE FROM:
CW COMPUTER WORKSHOP

LONDON & MANCHESTER AND MANY OTHER AGENTS THROUGHOUT THE UK & ABROAD

MAINTENANCE THROUGH COMPUTER FIELD MAINTENANCE LTD.	LEASING & INSURANCE THROUGH G. F. WHYBROW (LEASING & MANAGEMENT) LTD.

SOUTHWEST TECHNICAL PRODUCTS CO.,
38 DOVER STREET, LONDON W.1. 01-491 7507

Apple Bally DEC Horizon Imsai MSI
APF Cromemco DG ICL IBM NCR
Altair CDC HP Intercolor Kim OSI

Get the most out of your computer!

creative computing

#1 in applications and software!

PET PolyMorphic SOL-20 SWTPC Tektronix Vector Graphic
TRS-80 Sorcerer Siemens Video Brain Wizard Xitan

No computer magazine gives you more applications than we do! Games. Puzzles. Sports simulations. CAI. Computer art. Artificial intelligence. Needlepoint. Music and speech synthesis. Investment analysis. You name it. We've got it. And that's just the beginning!

Whatever your access to computer power—home computer kit, mini, time-sharing terminal—*Creative Computing* is on your wavelength. Whatever your computer application—recreation, education, business, household management, even building control—*Creative Computing* speaks your language.

Read through pages of thoroughly documented programs with complete listings and sample runs. All made easy for you to use. Learn about everything from new software to microprocessors to new uses for home computers. And all in simple, understandable terms. And there's still more. *Creative Computing* discusses creative programming techniques like sort algorithms, shuffling and string manipulation to make your own programming easier and more efficient.

We can even save you time and money. Our extensive resource section is filled with all kinds of facts plus evaluations of hundreds of items. Including microcomputers, terminals, peripherals, software packages, periodicals, booklets and dealers. We also give you no-nonsense equipment profiles to help you decide which computer is best for you—before you spend money on one that isn't.

We've got fiction too. From the best authors in the field, like Asimov, Pohl and Clarke. Plus timely reviews of computer books, vendor manuals and government pamphlets. And so much more!

Subscribe today and receive your copy direct from America. Avoid disappointment in the event your shop runs out and save time and money by subscribing. Please allow 8 to 12 weeks for subscription to begin.

Back numbers (Volumes 3 and 4) and collected annual volumes (1 and 2) are available; also *Basic Computer Games; Microcomputer Edition*, the best-selling book of computer games in the world. See coupon for prices.

Available in selected computer shops and by post.

I want to get the most out of my computer.
Please enter my subscription to:

creative computing

• 12 issues per year

☐ 1-year £13 ☐ 3-year £36
☐ 2-year £25 ☐ Lifetime £225

Back numbers, £1.75 each

☐ Vol 3, No 1 ☐ Vol 4, No 1
☐ Vol 3, No 2 ☐ Vol 4, No 2
☐ Vol 3, No 3 ☐ Vol 4, No 3
☐ Vol 3, No 4 ☐ Vol 4, No 4
☐ Vol 3, No 5 ☐ Vol 4, No 5
☐ Vol 3, No 6 ☐ Vol 4, No 6

Best of Creative Computing books, £6.95 each
☐ Volume 1 ☐ Volume 2

Basic Computer Games book, £5.50 each

Amount enclosed ____________

Name ____________

Address ____________

City ____________

Postal Code ____________

Send to:
Creative Computing, Attn: Hazel Gordon
Plot 23, Andrew Close, Stoke Golding,
Nuneaton CV 13 6EL

PERSONAL COMPUTER WORLD

UK 50p
US $2.00

Vol. 1, No. 10
February 1979

Europe's first magazine for personal computers for home and business use

CONTENTS

ISSN: 0142 0232

Editorial and Advertising Office:
62A Westbourne Grove, London W2
Phone: 01 - 229 5599 (Publisher)
01 - 727 8758 (Editorial and Advertising)

Publisher:
A Zgorelec
Editor:
Meyer N. Solomon
Editorial Assistant:
Roger C. Wilkins
Policy Advisor:
Peter Crofton-Sleigh, FRAS

Consultants:
John Coll, Mike Dennis, Neil Harrison, Charles Sweeten, Patrick Sutton, Michael James, R. W. Davy, David Hebditch, Sheridan Williams, Stephen Castell
Art: **Sauveur Laurent Sant, Kathryn Hamme**
Secretarial:
Vanessa Blackburn Kiddle

Layout Consultants:
D. Norris, T. Gabos
PCW Photography: **Yoshi Imamura, Peter McGee**
Typesetting & Artwork: **Gilfillan**
Cover: The Attache

CONTRIBUTORS:

We welcome interesting articles written simply and clearly. You need not be a specialist to write for us. MS should not be more than 3000 words long, lines double spaced, with wide margins. Line drawings and photographs wherever possible. Enclose a stamped self-addressed envelope if you would like your article returned.

Manufacturers, suppliers and dealers are welcome to contribute technical articles, and send product information, but we are pledged to an independent viewpoint and will publish evaluations and reasoned criticism or praise, space permitting. Naturally there will be right of reply. Views expressed in articles are not necessarily those of Personal Computer World.

We may make arrangements to offer our readers products at special prices, for a limited period, in line with the policy outlined above.

Published monthly by Intra Press, 62A Westbourne Grove, London W2. Phone 01-229 5599. Contents fully protected by copyright. All rights reserved. Subscription rates: Britain £8 for 12 issues. Prices include postage. USA — $10 for six issues, $20 for 12 issues. Continent and elsewhere £9.80 for twelve issues. Prices include postage.
Printed by Carlisle Web Offset, 55 Conduit Street, Newtown Trading Estate, Carlisle CA2 7NR. Sole UK Distributors: Seymour Press Ltd., 334 Brixton Road, London S.W.9., England. Distribution to specialist shops by Intra Press.

Editorial

Young People

Back in the days when I was a committed chess player I once graciously sat down to give a twelve year old boy a lesson, or perhaps two, in chess. Three games and three defeats later I staggered from my chair wondering what had hit me.

Duncan Willis (see photo) exhibited his 77-68 at our show. He built the whole thing himself, making only one call to Tim Moore at Newbear.

Duncan Willis shows off his 77-68 Micro at the **PCW** *Show.* Photo: **Nobby Clark,** The Observer

Mark Colton (see "Punchlines") was seventeen at the time he wrote his articles for **PCW.**

Paul M. Jessop is now a veteran writer; he's eighteen or thereabouts.

Articles to come: "Checkmate", by **Francis Best,** aged eighteen; "Sixteen Bits of Power", by **Caspar Bowden,** aged seventeen.

I'm delighted. I sincerely think that the word "world" in **PCW** is all important – it means accommodating (though not necessarily endorsing) varied interests, opinions, levels of knowledge and age groups. (I have an article by a seventy-eight year old coming up.)

My only anxiety (and that's not too strong a word) is that we will not receive articles by women. I would like to issue a strong invitation to women readers to contribute to **PCW.**

Apology

The article "Binary Conversion" in the January issue was wrongly attributed: the author is **Leonard L. Leyrer,** to whom we offer our sincere apologies.

Publisher's Letter

Dear Reader,

This is our anniversary issue, I would like to thank everyone, readers and advertisers, for their support.

Acquaintances of mine in the publishing business wonder how we get such interesting readers' letters. I reply, "because we have interesting readers".

We're both praised and criticised by readers – but every single letter is a letter from a friend.

I have some great news. We have a stand at the *Microsystems 79* exhibition at the West Centre Hotel, Lillie Road, London, SW6 – the same exhibition at which we launched **PCW** last February. We're going to use this occasion to launch yet another publication – **Computertrader.** It will be full of trade news and information, and will be a monthly.

And in mid-March, if all goes well, there will be a *third* string to our bow. This will be a bimonthly magazine, **Computers And Small Business Applications.** No coincidence that its acronym is CASBA!

SUBSCRIPTIONS

When **PCW** started publication, we had a special six-issue offer. When these subscriptions expired, we sent out reminders.
The renewal rate was 70%!
PCW reader loyalty is becoming a byword in publishing. If you're having difficulty in obtaining **PCW** at your newsagent, take our subscription. You can find the details at the foot of P.3.

REQUEST TO READERS

Personal Computer World, for its future publishing projects, is now compiling a comprehensive list of magazines, periodicals and books on personal computers and microprocessor applications, as well as manufacturers, dealers and suppliers.

Any reader living in Europe who is able to help – please send your information to

PCW (Information Publications)
62A Westbourne Grove, London W2
England

Letters

PUZZLE DAZZLE – No. 2

I enclose a small puzzle for those of your readers that own *MK14* micro's. I hope you can find room in your magazine for it!

P.S. Anyone for a MK14 Users' Club?

MK14 Puzzle

A program occupies memory locations OF12 to OFFO.
A programmer decides to make the program loop back to OF12. He cannot use the registers P1, P2 or P3, and the program must jump directly from OFFO to OF12.
What coding must he write?

Geoff Phillips,
8 Poolsford Road,
London NW9 6HP

PCW *£5 goes to Geoff Phillips for setting the puzzle, and £5 each to first three correct solutions received.* **PCW**

Catch them young

As a teacher of Mathematical Sciences I am finding that pupils aged 14 - 15 with only a few weeks' experience of computing are avidly buying your magazine. The reason, apparently, is that although many of the articles are meaningless to them, there is always at least one program that they are able to run, whether it be a long program (Submarine Chase, Geography quiz, etc.) or a short one that they can experiment with – computer art, Pet preening and so on.

Buying your magazine is helping to educate my students with articles that, sooner or later, will become meaningful.

The point of this letter is that with the rank beginner in computing – keep up the good work with the programs.

D. Adams, (Ellis Guilford Comprehensive School)
367 Spring Lane,
Mapperley Plains,
Nottingham.

Vintage Whine

I have subscribed to PCW before I ever dreamed of actually *owning* a computer. Now that I do and I have tasted the joys and the frustrations of all that go with being awakened to this new and exciting world, I have something to say:

I don't think I should complain too much about the fact that the companies involved in marketing personal computers are pathetic in their lack of available peripherals. This is the way the British do business. They just happen to be fortunate to have caught a highly successful bandwagon. In the States these great entrepreneurs would last about 3½ minutes!

I don't really want to complain about the fact that there is a rip-off market going on in software that is over-priced, full of errors and mechanically faulty.

But what I do want to complain about are the snide remarks and whining negative attitude that oozes out when I mention that I bought my personal computer from Lasky's.

Until dealers realise the reality that there are thousands of prospective customers who cannot afford to blow hundreds of pounds in cash, they are going to have to be satisfied with a small, one-sided market.

Ron Singer,
07 Nelson House, Dolphin Square, London SW1V 3MY

PCW *A glance through the magazine shows that recently the situation has improved* **PCW**

Mislocated Title?

The article on computing in geography by J. D. and T. C. Lee in your November issue cannot be allowed to pass without comment. There would have been no objection if it was called 'Rote learning with a computer' and 'Program to test general knowledge'.

Only in a few schools is the type of geography described still taught. Instead pupils are taught how to look up information of that sort, should they ever need it, and the majority of school geography is spent in a study of the way in which society interacts with its environment in various parts of the world. Pupils are introduced to such varied topics as North Sea oil, its origin, extent, exploitation, and impact; to the problems of declining industrial regions, third world agriculture, and the growing range of urban and rural planning problems. (Computers are used as a teaching aid in all these topics).

The content of geography, and the skills taught within geography, now produce students able to understand major political and social issues without prejudice. It is a rapidly developing subject, changed out of all recognition since most of your readers left school. Because of this it is not surprising that the authors of the article were unaware of the changes. However, it is important that your readers do not get the impression that their program has any relevance to contemporary school geography.

One of the most striking features of contemporary geography is its very heavy use of computing, for data analysis, for the development and testing of explanatory models and simulations, and in automated cartography and a wide range of computer graphics.

Nor is this activity confined to universities and polytechnics. The Geographical Association has had a major research grant to investigate the use of computers for teaching purposes and now provides a range of fully documented programs designed for use in a teaching situation, programs which perform the tedious data management and low level calculations often necessary in contemporary geography but of no educational merit. It also acts as a centre for exchange of information and is in touch with about 400 teachers in secondary and tertiary education who are active in the use of computers.

We are now in an exciting phase when microcomputers are increasingly being used in education. Educators are researching the best machinery, and developing educational software that makes effective use of the power and availability of microcomputers.

Full details of the work of the Geographical Association are available from the writer.

David Walker, Lecturer, Dept. of Geography,
University of Technology, Loughborough, LE11 3TU

PCW *Call it what you will – children will still have* **fun** *with the program. Of course, that isn't the point of David Walker's letter, and we appreciate and agree with the Geographical Association's outlook.* **PCW**

Orient your Computer thisaway

May I make an appeal for assistance through your columns? Wrekin Orienteers is organising the 1979 Midlands Orienteering Championships. It will be held on Sunday 4th March in the Telford area of Shropshire, and will attract upwards of 2,000 competitiors of all ages from 10 to 60.

Ours is a wholly amateur sport and putting on an event of this size makes heavy demands on our resources. We would like to reduce this burden by using computer assistance in preparation of staff lists, programs and computing results.

Do any of your readers have a microcomputer system, with disc and printer facilities, which we could borrow, or hire at purely nominal rates, for a period of two months before and immediately after the event? If so, I would be pleased if they would contact me at this address/telephone number.

P. E. Walker, 9 Queens Drive, Newport, Shropshire TF10 7EU
Telephone (evenings) Newport (0952) 810060

Computer Quips

While it was good to see another attempt at solving the problem of "What to Do after You Hit 'Return' and it Aborts Your Program", PCW, November 1978, p.34, Mr. Smith inclusive – OR your staff must bear the responsibility for the numerous errors in the program shown:

Line 1000: insert a 'space' character between the two backspace arrows. **Line 1000:** "THEN GO TO" is redundant. **Line 1005:** In my PET and all others I have seen, SL is 48 and SU is 57. **Line 1010:** Delete the semicolon. **Line 1015:** Delete "+1". **Line 1100:** Delete "=". **Add line 1013:** "IF SC = 64 GOTO 1020". This will prevent the program from aborting on input of upper case characters or lower case characters other than numbers or the specified letters.

While on the subject of PET, here's a strong vote for the use of the following standard code in printed PET programs. Each is two-character, lower case.

sh: shift the next character. (alternately underline the char.) **sl:** shift lock. (hold shift down until sr). **sr:** shift release. **ho:** home. **cs:** clear screen. **cl, cr, cu, cd:** cursor left, right, up, down. **rv:** reverse. **ro:** reverse off. (alternately rl and rr for compatibility with shift code). **sp:** space. **in:** insert. **de:** delete.

The need for some such coding comes about because of PET's flexible cursor facilities and large vocabulary of special graphics symbols.

It's hard to believe, but there are still some surprises hidden in PET. Here are a few I doubt even the designers know about:
(Write one of these lines and then call for a listing.)

THE SHAKESPEAREAN ACTOR
```
4502spREMs1, #sr Bspsh0 spsh (spsh #Bs1+N:4sr
```

THE RELIGIOUS CONVERSION
```
26912spREMs1,←Q ←Q ←Q ←Q ←Q ←Q ←QsrD!
```

THE DIRTY OLD MAN;
```
11234spREMs1,F( $=/Z0? $Y $?+)sp;!"B'UVsr
```

If it doesn't make any sense, try verbalizing what's on the screen. If that doesn't help, check your program lines very carefully. Your PET is trying to tell you something!

Frank Chambers, Rock House, Ballycroy, Westport, Co. Mayo, Ireland.;

Incontinent?

'HOW TO CONTROL YOUR BASIC FUNCTIONS', Christopher Smith, (PCW November 1978) has taken a sledgehammer to crack an egg, with the suggested method for avoiding PET returning to command mode when the RETURN key is pushed in response to an Input request.

```
10 INPUT "ENTER YOUR AGE [2 RIGHT] *[3 LEFT] ";
   G$ :G=VAL(G$):IF G=0 THEN PRINT "[2 UP]":GOTO 10
```

is far simpler.

This line uses the program listing convention for PET from the American magazine *People's Computers.*

Whenever square brackets appear in the listing, neither the brackets nor the text they enclose should be typed literally. Instead, the text between the brackets should be translated to keystrokes. For example, [2 RIGHT] means press the second CRSR key twice.

John Collins, 90 Charing Cross Road, London WC2H 0JE

Programmed Profits

I can only conclude from the somewhat erratic delivery of my copy of PCW, and the complete failure of the November issue to arrive at all, that your computerised system for handling subscriber lists (surely you must have one, with a name like PCW) has not been introduced to software quality assessment techniques. Or was it that the programmer found that he could save the company a lot of money by issuing only 90% of the copies, and blaming the rest on software errors. This opens a whole new field of business economics!

May I congratulate you on your sortie into the problems of Assembler writing, though I would have liked to have seen an introductory article covering the general principles of syntax decoding. Also, how about an occasional page or two of useful routines (e.g. 16-bit or floating point multiply etc.), along the lines of the technical tips featured in most electronics magazines?

J. R. Keneally, 31 St. Helens Road, Weymouth, Dorset DT4 9DY

PCW *This ingenious approach to increased profits hadn't occurred to us, but now . . . . .* **PCW**

A PET tip

Here is a tip for PET 2001/8 users who find themselves inputting a null string to an input statement. PET recognises the keyboard as device number 0. The screen as device number 3.

```
10  OPEN 1,0,0
20  INPUT #1,A $
30  IF A$ ="END"THEN 100 (OR, IF A$= " " THEN 20;
    INSTEAD OF IF A$= "END")
40  INSTRUCTIONS FOR WHAT TO DO WITH A $ GO
    HERE THEN HAVE A GOTO 20 TO RETURN TO LINE
    20
50
60
70
80
90
100 CLOSE 1
```

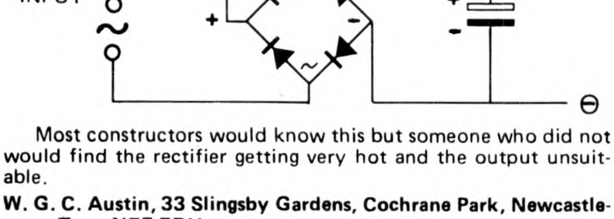
(Line 30 allows you out of loop).

No question mark will appear for the input statement, but as long as a string variable is used a null input will not cause the program to return to command level.

Thomas Turnbull (Petsoft Consultant), 49 x 9th Row, Ashington Northumberland NE63 8JY

Correction to a correction . . . infinite regression?

Re Personal Computer World, November 1978, p. 39 – your corrections to July Punchlines are wrong in one diagram. The rectifier circuit as given would be extremely inefficient if it worked at all! – It should be:



Most constructors would know this but someone who did not would find the rectifier getting very hot and the output unsuitable.

W. G. C. Austin, 33 Slingsby Gardens, Cochrane Park, Newcastle-upon-Tyne NE7 7RX

PCW *The editor is at the moment in a monastery, being scourged* **PCW**

Zeal for the Z80

You *must* ask Mike Dennis to stop making vague promises to "get around" to designing a Z80 based system! Z80 architects are conspicuous by their absence; having half-volunteered for the task, perhaps *he should be sponsored by PCW to produce* the best Z80 based home system to date.

Looking at other projects from the early growth-period of British personal computing we can learn from the experience and criticism which they have raised.

The first point is: that anyone choosing a 6800 or 8080 as the heart of a system is patently "chickening-out" of the use of the slightly-more-daunting Z80. Regardless of the claimed comparative ease of using the other devices, it must be obvious that the 158 - Instruction-Set Z80 offers greater ultimate versatility and the possibility of tighter programming – particularly in its efficiency when applied to compilers, interpreters and assemblers, and probably also in operating systems.

Before Mike Dennis crawls into his den for a lengthy stint at his self-imposed task, could I start the ball rolling by 'chipping-in' with a skeletal basis of the system I envisage, and hope for comment from other readers which can further assist our designer as he settles to his monastic task?

The circuit should – unlike some – provide for full use of the *whole* Z80 Instruction Set. Circuitry should be chosen for cost effectiveness, and need not be restricted to the Z80 'family' of chips. Whether or not to use a 'conventional' bus-structure is a moot point; ribbon connectors can, in many applications, prove both more convenient and reliable, *and cheaper* than a bus-and-motherboard structure. 'Chassis-bashing', in the earlier days of electronics, was reasonably cheap and effective, and present indications are that it still is!

The power unit should be generous in capacity.

Memory should be at the choice of the user: either static or dynamic-with-refresh, as alternatives. This should also take into account a reasonably generous amount of ROM for a good (and efficient) BASIC, providing three levels of mathematics: integer, six-digit and some larger number of digits; six digits can be common to both the integer and a simple floating-point system, while the other system should leave no-one critical of the machine as a number-cruncher.

An efficient monitor, editing capacity and **operating system** should be provided; CP/M is rapidly taking over as a standard, and can hardly be left out of this system.

A full keyboard, with the proper feel, should be specified. It should provide all 128 characters, and could have additional 'shift' for graphics if desired. There should be provision for a numeric key-pad for anyone wishing to add it as an extra, as should additional keys for some of the more frequently used functions.

Screen display should be either VDU or TV-modulated, and should provide 80 characters per line for the very practical reason that this is the width of an A4 page; A4 can be very useful in print-out, either in single sheet form, or in 210mm rolls. In office applications it lends itself readily to copying and filing systems and is a business standard which cannot be overlooked. Regardless of noises to the contrary, a good modulator can quite capably support an 80 character line with good definition; perfectly good TV's are now available on the second-hand market at £8 to £12, and good modulators of adequate bandwidth are not expensive – too good an opportunity to miss!

As a teaching aid, the system should provide both an LED-based single-step-and-examine facility, and also single-step screen display as in G. J. Flanagan's September article: "The Soft Facade". One could hardly conceive of a better de-bugging facility or instruction method.

Using the 80 c.p.l. display, an early program for **Word Processing** should be made available. This should be modular, insofar as it should support an automatic correction and edit facility, but in its simplest form this should be a single page program; the exotic multi-page version could be added later. Reason for asking for this facility is to provide a sophisticated standard typewriter capability; many wives are competent typists, and as their financial support is essential to many purchasers it is only fair that they should be given a crack at the apparatus. It is also a good advertising gimmick with which cap-in-hand husbands could obtain their hearts' desire.

Commercially, a tie-up with a major electronics company would be sensible: it leads to lower first cost and better back-up facilities. The larger the company, the better the prospect.

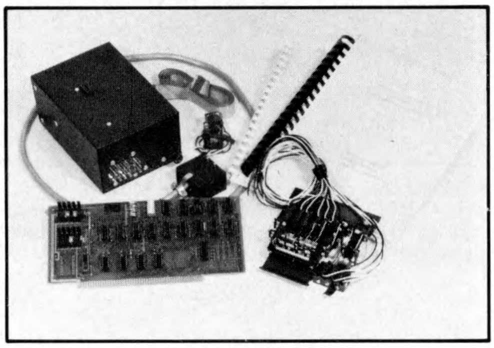
B. A. Martin, 99 Northdown Road, Solihull, West Midlands B91 3ND

PCW *We would not undertake a PCW system unless we were sure of guaranteed quality and an absolutely cast iron back-up service* **PCW**

READERS

Using a SORCERER, IMSAI, MSI6800, HORIZON, VECTOR MZ . . . . or any other small system?
Write to us.

Tid Bits PRODUCTS . . . . COMPANY NEWS . . .



Universal interface unit for IBM Selectric

ESCON announces the development of a universal interface unit for its IBM selectric typewriter conversion system. The present unit allows any microcomputer with an S-100 bus to output to an IBM selectric typewriter. The new unit will interface to any RS-232, IEEE-488 or parallel port. A microprocessor is included on the circuit board for data flow control, formatting and character set selection. The unit allows for interfacing to a wide range of computers such as the TRS-80, Apple and Sorcerer.

The installation on the selectric is easy, takes only several hours, and does not affect normal typewriter operation. For those who do not want to convert their own typewriter ESCON provides factory installation service or can recommend local qualified computer stores throughout the U.S. Selectric typewriters with conversion systems installed in accordance with factory instructions are still eligible for IBM warranty and service provisions.

The addition of the universal interface capability greatly expands the number of computers for which selectrics may be used as output printers; for example, TRS-80, Apple, Sorcerer, etc.

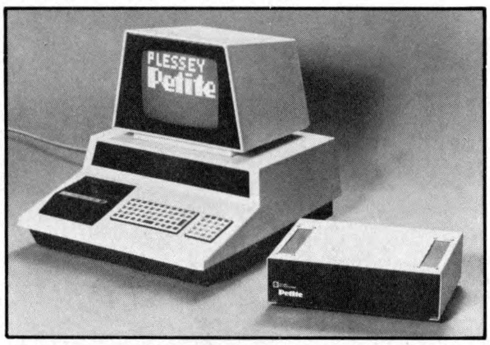
For further information contact:
ESCON Products, Inc.,
171 Mayhew Way, Suite 204,
Pleasant Hill,
California, 94596.
Telephone: (415) 985 4590

MICRO-BASED CONTAINER HANDLING SYSTEM

Software Architects has obtained a contract with Multiterminals (Rotterdam) b.v. for the computerisation of its container handling system located in Rotterdam Harbour. The system, when completed, will enable all eleven of Unitcentre's 200 metre high cranes to be positioned within 10 metres of each other under the direct control of a centralised Siemens computer.

Within each crane will be a Zilog MCZ micro computer which will process sense-data to obtain the crane's position and thence pass it to the Siemens computer via a high speed serial communications link. In addition the MCZ will also control a display in the crane driver's cab showing the exact position of the crane and the identification number of the container which is to be moved.

For further information contact:
Leslie Dewhurst on 01 - 734 9402 or write –
34-35 Dean Street, London W1V 5AP

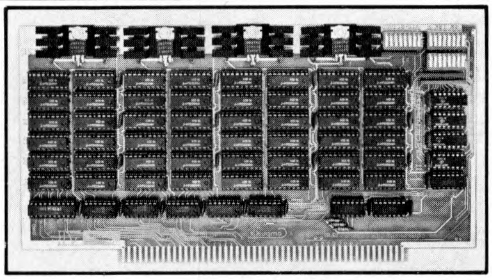


Petite, *the Plessey Microsystems 24K byte add-on memory that expands a Pet personal computer to its full addressing capacity.* **Available from TORBUS, 500 Chesham House, 150 Regent Street, London W.1. Telephone: 01 - 734 5351**

Another Newbear Fact

A VDU, Model 700, from **Newbear** is low cost, offering: upper case ASCII; 64 Ch x 16 lines; scrolling; full cursor control; RS232C/V24; 110 Baud, 300 - 1200 Baud; 12" display; separate keyboard; full duplex or half duplex. Price is £299.

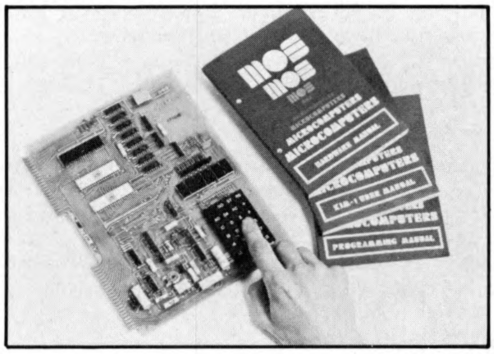
Further details and demonstrations from nearest Newbear Store or write: 7 Bone Lane, Newbury, Berks. RG14 5SH.
Telephone: (0635) 49223



LTT Electronics, a mail order only outfit, can offer some of the famous Godbout computer products, including the best selling Economoram (TM) range of memory boards. For details of range and prices, write to 37 Orlando Road, London S.W.4.

Brian Reffin Smith (see **PCW** Vol. 1, No. 8) can offer his "Intelligent Programs" – art/conversational programs on tape for the Research Machines and Pet. He can also give advice on Graphics, and Basic listings of programs he has created. Write to him for full details and catalogue at 32 Kensington Park Gardens, London, W.2.

Microdigital of 25 Brunswick Street, Liverpool now claims to have the biggest and most comprehensive list of "readware" – books on personal computing. Also its own brand of coding pads and high quality cassettes. Details from **Bruce Everiss.**



The KIM 1

Price Breakthrough

KIM 1 is now £99.95. Fully assembled, 6502 microprocessor, 2K bytes of ROM, 1K of RAM, Keyboard, 6-digit LED display. Full documentation. Expansible. Marketed by the innovative *G. R. Electronics* of Newport, Gwent; and *Marshall's* the well known electronics distributors with shops in London, Glasgow and Bristol.

G. R. Electronics also offer a £475 IBM 'golfball' based printer for PET.

Home in on the Texas range and Fly with OSI.

Abacus Computers of 62 New Cavendish Street, London, announce that Texas Electronic Instruments have appointed them worldwide distributors for their range of microcomputers.

Abacus are also sole distributors of Ohio Scientific Instruments' range of computers, including the Superboard II. Dealer and general enquiries to Derek Rowe at above address.

NASCOMpatible

JWM Electronics, 60 Balcombe Street, London N.W.1 (01 - 262 2936 *or* 01 - 402 9244) has jumped on the NASCOM bandwagon with three kits. KIT 1 is for Alphanumerics, Graphics; KIT 2 is a Graphics RAM with colour decoding, R.F. modulation; KIT 3 a programmable sound F/X Generator. Write or ring for details.

The ICL Gambit
International Computers Ltd., has rescued the famous Hastings International Chess Congress from oblivion.

This public spirited move means that a vital event in the chess world will continue to foster new talent.

The Almarc of Quality
Almarc Data Systems Ltd., have introduced time sharing on the Vector Graphics Z80A microcomputer, and state that as a result the system is ideal for schools. The computer uses the S100 bus standard and so can accommodate a vast range of add-ons such as video graphics boards, music synthesiser, voice recognition, and a real time clock. Free advice and a brochure from 29 Chesterfield Drive, Burton Joyce, Nottingham. Telephone: 0602 248565.

TRITON makes Waves
The Triton hobby microcomputer, designed by Mike Hughes, now has over 400 users. This single board system is expandable to 64K, and has BASIC, 56 key ASCII keyboard, 256 I/O ports among its features.
Full details from:
TRANSAM Components Ltd.,
12 Chapel Street, London.
Telephone: 01 - 402 8137
Or write for a catalogue, with s.a.e.

Teletext/Viewdata from Technalogics
A 6800 microprocessor based decoder with "powerful local computing" facilities – the TECS – is now on the market.

The system is crammed with features, some of which are: program access to Teletext information such as share prices; all colour display facilities inherent in Teletext and Viewdata; ROM -resident TECS mini-Basic; machine code monitor program.
Full details from:
Technalogics,
8 Egerton Street, Liverpool.
Telephone: 051 - 724 2695

Mini Micro, specialists in PET games, now offer books from leading American publications in the computer field. Readers can obtain the Books and Games Catalogues by writing to 47 Queens Road, London, N11 2QP.

New American Micro Magazine: A monthly publication devoted to the Motorola 6800, *'68' Micro Magazine* is intended to be objective, giving equal space to criticism and rebuttals.
Details from:
Hamilton Publishing Inc.,
3018 Hamill Road,
Hixson, Tn., U.S.A.

Pelco (Electronics) Ltd. draws attention to the **Rockwell** R6500 – a family of 10 software - compatible CPUS, eight I/O, ROM, RAM and one-chip memory – I/O - timer circuits operating at 1MHz or 2MHz speeds with a single 5V power supply.

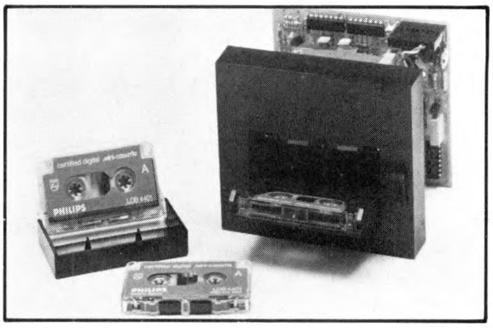
In Rockwell's range is the SYSTEM 65, a floppy-disk based, "powerful yet low cost" complete development system.
Contact:
Pelco (Electronics) Ltd.,
83 - 85 Western Road,
Hove, Sussex BN3 1JB
Telephone: (0273) 722155

A new Source of software
Source is a new software company started by two ex-employees of Southwest Technical products (UK).

In the future it will be providing considerable support for the leading microcomputers in the form of powerful system software.

It will also contract to write systems and applications programs for whoever might require them, and interface peripherals which haven't already been connected.
Contact:
Source, 12 Vivian Road, Wellingborough, Northants.
Telephone: (0933) 224040

Research Resources Ltd., offer a statistical package for **SWTP** compatible microsystems. The package – named SAM (Statistical Analysis for Micro-computers) – requires a minimum configuration of 32K and a dual floppy disk (mini or standard). The current version contains fourteen analyses; such as *edit, histogramming, regression* and *T-Tests.*
Contact:
RESEARCH RESOURCES Ltd.,
P.O. Box 160, Potters Bar, Herts., England.
Telephone: Potters Bar 54737

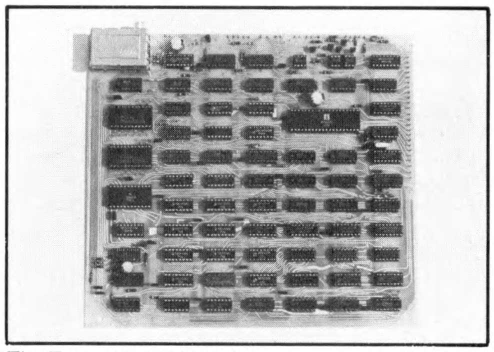


New from Philips
Philips' new M-DCR series of Mini-Digital Cassette Recorders provides 128 kbytes of serial memory, recorded on two tracks of interchangeable certified digital Mini-cassettes.
Contact:
V. L. Drayton, M.E.L.,
Manor Royal,
Crawley, West Sussex RH10 2PZ
Telephone: 0293 32850

The Tasteful Tangerine – VDU Kit
The TANGERINE 1648 is a fully self-complementing module enabling the transfer of data to and from any computer, albeit micro, mini etc., and linking the intelligent unit with a video display terminal. This can either be a video monitor or a conventional television set that has a UHF tuner and operates at 625 lines with a field rate of 50 Hz.

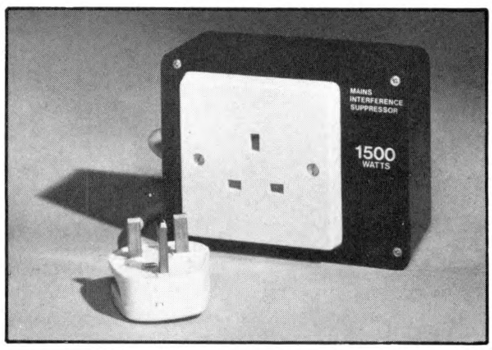
The TAN 1648 generates 16 rows by 48 characters. Non-interlaced scanning is used with an active line width of 48μ Secs. Should this be too wide, even after adjusting the width control of the receiver, the left hand margin may be pulled in by 4 characters thus reducing the line width to 44 characters which with this VDU kit produces an active line of 44 μSecs.

Other features: Built in repeat function and lower case function, Composite video output, decoded character control outputs.
Full details:
Tangerine Computer Systems Ltd.,
Rivermill Lodge, London Road,
St. Ives, Huntingdon, PE17 4BR



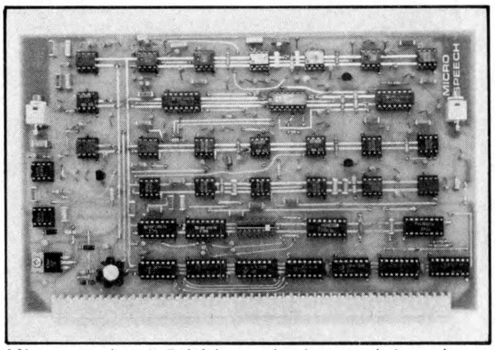
The Tangerine VDU board

Beyts Logic Ltd., are marketing a plug-in "Suppressor" for systems. Combats mains-borne interference and helps safeguard data integrity. Comes complete with 65 cm lead, 13 amp plug (fused for 7 amp) for £17.90.
Details from:
Beyts Logic Ltd.,
Windmills Road, Sunbury, Middx.



NEW PRODUCT – MADE IN THE U.K.

MICROSPEECH is a microprocessor peripheral that produces synthetic speech. The card containing all the electronics plugs into the standard SS50 bus on the SWPTc and MSI 6800 microcomputers. The software translator program (MSP2) converts phonetic code (which is similar to normal spelling) into sets of data that control the speech synthesiser. The data, when decoded, produces nine control paramaters which determine pitch, amplitudes, and resonant frequencies in the speech model. What goes in are phonetically spelt phrases, and what comes out is synthetic speech.



Microspeech: A British synthetic speech board.

The speech model is a three format synthesiser with separate nasal and fricative branches. A digital noise source and a voltage controlled oscillator produce the signals that drive the unit. Alternatively an external signal may be fed in and articulated, making speaking musical sounds readily attainable.

MICROSPEECH is for the microcomputer owner, and a useful tool for those involved in speech research, education and system design.

As well as the standard phoneme translator package, a disc based BASIC interpreter with speech output is available as a software option.

The software is available on floppy disc or cassette.

Contact:
Costronics Electronics,
13 Field Heath Avenue, Hillingdon, Middx; or
Tim Orr, 55 Drive Mansions, Fulham Road, London, S.W.6

Crystal Clear

Torquay has had for some time, a Micro-computer and Components shop – CC/Crystal Electronics – run by a team of hardware and software engineers, headed by Trevor F. Brownen.

Among their main computer producrs are the Apple II, the Nascom I, the Newbear 77/68 System and the Atari Video Computer System, and they are at present evaluating many other products.

The shop is open every day from 9.30 a.m. to 6.00 p.m. except Wednesdays and Sundays.

For advice or help telephone: 0803 22699.

TRS-80 Software is now available from A. J. Harding of Bexhill. This includes all types of programmes for the TRS-80, ranging from games to business software. Mr Harding, the U.K. director of J. & J. Electronics Ltd., Canadian mail order semiconductor distributors, has available both programs imported from the U.S.A. and programs from English authors. Having been involved with Micro-processors from their inception in North America and being one of the earlier purchasers of TRS-80 equipment, Mr Harding is in a good position to bring to this country a good assortment of software. An SAE will bring you his list:
A. J. Harding, 28 Collington Avenue, Bexhill-on-Sea, E. Sussex.
Telephone: (0424) 220391

EXPANDING YOUR HORIZON

Users who have felt that the maximum disc storage of 270KB, i.e. three minifloppy Shugart drives of 90KB each, was insufficient for their needs will now be pleased to know that the North Star Horizon, marketed by Equinox Computer Systems, now supports four double density drives in place of the former three single density drives.

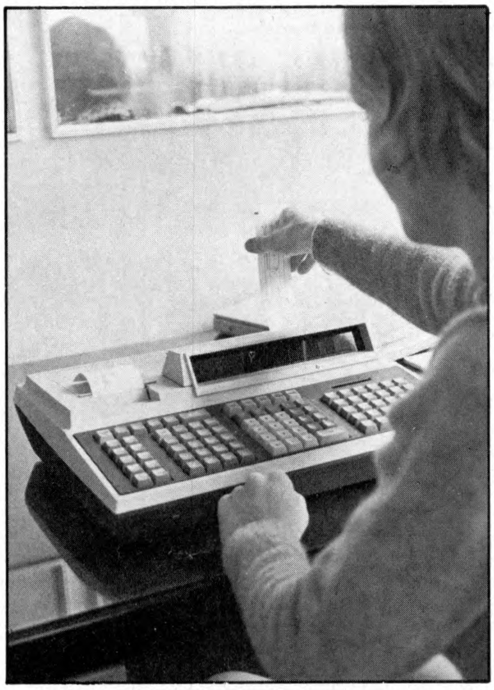
Both the powerful BASIC and DOS have been upgraded to accommodate the increased capacity. Application software will continue to operate with little or no change. Release 5 of the North Star Basic is being issued at the same time.

Further information from:
Mike Kusmirak,
Equinox Computer Systems Ltd.,
32/35 Featherstone Street,
London EC1Y 8QX
Telephone: 01-253 3781/9837

Payroll package "boosts TI personal-computer sales"

A payroll package developed for the Texas Instruments **SR60A** personal-computer/calculator by a Nottingham business equipment firm, **Betos Systems Ltd.**, has enabled the company to sell 50 of the machines to local businesses during its first year of operation. Betos believes that the SR60A, with its combination of printout, display and ease of operation, coupled with the payroll package, provides a versatile tool for managements of small to medium-sized businesses.

Contact:
Leonard Gelblum or Norman Burley,
155 Mansfield Road, Nottingham.



The payroll package developed by Betos Systems for the SR60A.

Crystal Electronics Shop

THE NEWBEAR COMPUTING STORE

7 Bone Lane, Newbury.

77 - 68

The Best Supported Hobbyist 6800 System in the U.K.

Available as Bearbags or as Individual Components.

Each Bearbag contains all the Components necessary to build a vital part of a microcomputer system. Each part is backed with the support necessary for such a complex project.

BEARBAG 1 :	77–68 CPU PCB and Components	£45.00
BEARBAG 2 :	77–68 LED's and Switches	£14.95
BEARBAG 3 :	77–68 Power Supply	£17.95
BEARBAG 4 :	77–68 5u Rack and Backplane	£27.70
BEARBAG 5 :	77–68 4K Ram PCB and Components	£75.00
BEARBAG 6 :	77–68 Mon 1 PCB and Components	£50.70
BEARBAG 12 :	77–68 VDU PCB and Components	£69.50
BEARBAG 13 :	77–68 Mon 2 PCB and Components	£t.b.a.
BEARBAG 16 :	77–68 EPROM Board and Components	£t.b.a.
BEARBAG 17 :	77–68 Interface Board (PIO) and Components	£t.b.a.

* TOUCH KEYBOARD *

- Full ASCII
- Assembled & Tested
- One Year Guarantee
- Comprehensive Handbook
- Touch Sensitivity Control

£37.50
inc. VAT and P&P

SYM·1

(Formerly VIM-1)

The new 6502 Micro from Synertek. Fully assembled and tested.
Ex. Stock. @ £199.00
+8% VAT, carriage £1.00

Floppy Disc Drives

5½"	SHUGART SA400 DISC DRIVE	£225.00
	Carriage by Securicor 'C' Rate	£4.50
8"	DRI SINGLE SIDED DOUBLE DENSITY	£325.00
	Carriage by Securicor 'C' Rate	£5.00
8"	DRI DOUBLE SIDED DOUBLE DENSITY	£365.00
	Carriage by Securicor 'C' Rate	£5.00

NASCOM·1

The Z80-Based Microcomputer Kit.
Ex. Stock. @ £197.50
+8% VAT, carriage £1.00
3 Amp Power Supply £24.50

Send for our New Winter Catalogue

OTHER BEARBAGS AVAILABLE

BEARBAG 7 :	4K Ram PCB and Components (Exorcisor Compatible)		£71.50
BEARBAG 8 :	8K Ram PCB and Components (Exorcisor Compatible)*		£16.00
BEARBAG 9 :	Petitevid VDU Kit. V24/RS232C interface. 64 Characters by 16 lines. Requires +5 –12 power supply, ASCII encoded. Keyboard, Video monitor of UHF Modulator (Bearbag 11) and a domestic T.V.		£85.00
BEARBAG 10 :	Kansas City Cassette Interface		£18.95
BEARBAG 11 :	UHF Modulator	12½% VAT	£ 4.50
BEARBAG 14 :	2708 Prom Programmer Kit (6800)		£35.00
BEARBAG 15 :	Promverter. Mikbug to 2708		£ 8.50
BEARBAG 18 :	Cottis Blandford Cassette Interface		£17.25
BEARBAG 19 :	TRS80 Level II 4K to 16K Byte Conversion		£85.00
BEARBAG 20 :	TRS80 Level I 4K to 16K Byte Conversion		£85.00
BEARBAG 21 :	Lower Case Kit for Petitevid.		£10.50

Postage and Packing 50p
*Postage and Packing £1.00
Add 8% VAT to all prices unless otherwise stated.

BEARBAGS ARE AVAILABLE FROM:–

NEWBEAR COMPUTING STORE
Bone Lane, NEWBURY. (Mail Order & Callers) 0635 49223

NEWBEAR COMPUTING STORE
2A Gateley Road, Cheadle, (Callers Only) 061 491 2290

MICRODIGITAL
25 Brunswick Street, LIVERPOOL. Tel: 051-236 070

MICROBITS
34b London Road, Blackwater, CAMBERLEY, Surrey.
Tel. CAMBERLEY 34044

CRYSTAL ELECTRONICS
40 Magdalene Road, TORQUAY. Tel: 0803 22699

THE BYTE SHOP
426/428 Cranbrook Road, Gants Hill, ILFORD, Essex.
Tel: 01-554 2177

Ins 1771 (Floppy Disc Controller)	£54.00	6830L/7 (MIKBUG)	£13.65
SFF 96364	£11.75	6800P	£ 7.15
4116	£11.50	6820*	£ 4.63
2114	£ 8.25	6840	£12.72
2102L-1	£ 1.35	6502	£14.93
2708 EPROM	£ 8.50	6522	£ 6.58
74S262	£13.70	6532	£12.56
		SWATBUG	£16.00

Books

77-68 Design Manual	£ 7.50	.50p
Introduction to Microcomputers.		
Volume 0: The Beginners Book	£ 5.95	.50p
Volume 1: Basic Concepts	£ 5.95	.50p
Volume 2: Some Real Products June 77 rev.	£11.95	1.00
8080A/8085 Assembly Language Programming	£ 6.95	.50p
6800 Assembly Language Programming	£ 6.95	.50p
Some Common Basic Programmes	£ 5.95	.50p
6800 Programming for Logic Design	£ 5.95	.50p
8080 Programming for Logic Design	£ 5.95	.50p
Payroll with Cost Accounting in Basic	£ 9.95	1.00
Understanding Microcomputers and Small Computer Systems	£ 7.56	.50p
Scelbi 6800 Software Gourmet Guide and Cookbook	£ 7.95	.50p
Scelbi 8080 Software Gourmet Guide and Cookbook	£ 7.95	.50p
The Scelbi Byte Primer	£ 9.95	1.00
The 8080 Programmers Pocket Guide	£ 2.35	.30p
Microprocessors C201	£ 8.00	.50p
Microprocessor Interfacing Techniques	£ 8.00	.50p
Guide to SC/MP Programming	£ 3.75	.50p
SC/MP Assembly Language Programming Manual	£ 3.75	.50p
SC/MP Microprocessor Applications Handbook	£ 3.75	.50p
Micro – The 6502 Journal	£ 1.70	.30p
Sym Reference Manual	£ 7.50	.75p
Sym Programming Manual	£ 7.50	.75p
First Book of Kim	£ 7.50	.75p
Kim 1 User Manual	£ 7.50	.75p
6500 Hardware Manual	£ 7.50	.75p
6500 Programming Manual	£ 7.50	.75p
101 Basic Computer Games	£ 5.50	.50p
Games, Tricks and Puzzles for a Hand Calculator	£ 2.49	.50p
Games with Pocket Calculators	£ 1.75	.30p
Star Ship Simulation	£ 5.10	.50p
Beginning Basic	£ 2.95	.50p
Introduction to Basic	£ 1.95	.50p
Some Common Basic Programmes	£ 5.95	.50p
Payroll with Cost Accounting In Basic	£ 9.95	1.00
Instand Freeze and Dried Programming in Basic	£ 4.95	.75p
My Computer Likes me When I speak in Basic	£ 1.65	.30p
Computer Programmes that Work	£ 2.40	.75p
What you do After You Hit Return	£ 7.00	.75p
6800 System Design Data	£ 2.00	.50p
Beginning Basic	£ 4.95	.50p
Learning Basic Fast	£ 6.30	.50p
8080A Microprocessor Family Quick Reference	£ 1.50	.50p

ALL MAIL ORDERS TO NEWBURY PLEASE

24 Hour Turnaround on Orders
Barclaycard and Access welcome, No Minimum. Overseas orders issued with a Pro-forma invoice. Minimum Official Order £10.00. Send for catalogue to Newbear Computing Store, Bone Lane, Newbury, Berks. Callers welcome Mon. – Sat. 9.00 – 5.30, *But please phone us first* on 0635 49223. New Office – 2A Gateley Road, Cheadle, Cheshire. Tel: 061 491 2290.

Begin.. A Simple Outline of Computing ..End

Brian Darling

With the continuing downward trend in the cost of computer hardware, many people are now taking their first serious look at computers. But confronted with RAM and ROM, BYTE and BASIC they simply do not know where to start. There is a danger that some will be so intimidated by the difficulties before them that they will give up, which would be a pity. This article sets out to show the complete beginner the outline of computing, leaving him to fill in the details himself. Using a conceptual model, the operation of the computer is explained and a program is devised to read in, add together and print the sum of, two numbers.

The Computer

A computer consits of five main units. Three of these; store, arithmetic unit and control unit, together form the central processing unit (C.P.U.). The fourth, is an input device — which for our purpose could most conveniently be a keyboard, similar to that on an electric typewriter. The last is an output device — which could be a printer, but on a personal computer is more likely to be a television screen.

Figure 1 shows the five units.

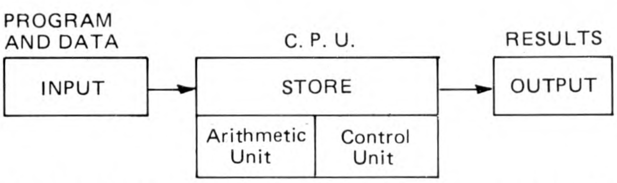


Figure 1. Notice that input goes into store and that output is taken from store.

The function of the input and output devices should be self-evident. The arithmetic unit, as its name suggests, performs the arithmetic and is quite similar to an electronic calculator. Store — which can be imagined as a set of numbered pigeon-holes — is used to store both the data and the program. The fifth unit, control, controls the overall operation of the computer and ensures that the program is executed one step at a time.

Rather surprisingly, this incredibly simple model will allow you to understand how the most complex programs are run.

The Program

A computer program is a list of instructions which cause a computer to carry out some task. Possibly to guide a space craft to the moon, but more likely to perform some rather mundane clerical job.

As an illustration of what programming involves we will show you how to write a program to add two numbers together. We will assume for the moment that the two numbers are already in store — which, you may remember, we visualize as a set of pigeon holes. In order to add the two numbers, they must be transferred to the arithmetic unit.

The program consists of just three instructions. The first transfers one of the numbers from store to the arithmetic unit. The second transfers the other number to the arithmetic unit, adding it to the first. The last instruction transfers the sum of the numbers back to store. This can be rather confusing at first but it should become clearer when we describe how the program is executed.

The three line program is shown below:

```
LOAD 103
ADD 127
STORE 107
```

The three numbers refer to three locations in store. 103 and 127 hold the numbers that are to be added and 107 will be used to store the result. The locations used are quite arbitrary and any others could be used, providing we first load into them the numbers we wish to add.

Be sure that you understand that it is the contents of store locations 103 and 127 that are added, not the numbers 103 and 127 themselves.

Execution of the Program

Before we can run the program we must enter it, together with the data, into store. This can be done by typing it on the keyboard. We will put the three lines into locations 001, 002 and 003. The program adds together the contents of storage locations 103 and 127, so we must enter into these locations the numbers, or data, that we wish to add. Say, 5 into 103 and 9 into 127.

Execution of the program is controlled by the control unit which has to fetch each instruction from store and execute it. This is known as the fetch/execute cycle. Part of the control unit is a counter which ensures that instructions are executed in the correct sequence. This unit is called the sequence control register (S.C.R.) or program counter.

The first instruction is fetched from location 001 and executed. It is:

```
LOAD 103
```

so the contents of 103 (5) are loaded into the arithmetic unit.

The second instruction is fetched from location 002 and executed. It is:

```
ADD 127
```

so the contents of location 127 (9) are added to the contents of the arithmetic unit (5) making the new contents of the arithmetic unit 14.

The third instruction is fetched from location 003 and executed. It is:

```
STORE 107
```

which transfers the contents of the arithmetic unit to store location 107.

2

Although this program is very simple it does illustrate how data is held in store and how it can be transferred to the arithmetic unit to be added, multiplied etc.

Machine Language

The above program is written in assembly language and a special program, called an assembler, is used to translate it into a form that the computer can recognize, called machine language. For engineering reasons computers use binary arithmetic, which is a kind of arithmetic employing just two symbols; 0 and 1. Machine language instructions consist of groups of eight binary digits, for example, the ADD instruction for one computer we know of is 00000010.

A binary digit is called a bit and eight bits together form a byte. Personal computers use a word length of one byte.

Machine language can be quite confusing to the beginner. But don't let this put you off, as your first attempts at programming will almost certainly be in a language called BASIC which we will describe shortly and you should find this much easier. Actually, machine language is not so difficult as it looks and you may get to quite like it later.

High Level Language

You may have been surprised at the amount of work involved in adding a couple of numbers together and, in fact, there is an easier way. High level languages allow instructions to be written in a form quite close to English. The three line program used above can be condensed to a single line in BASIC – which is the most widely used language for personal computers.

The BASIC instruction is shown below:

```
LET C = A + B
```

The biggest advantage of using a high level language is that it is no longer necessary to keep track of the store locations used. A program called a compiler, or interpreter, translates each BASIC instruction into machine language and also allocates storage locations to each of the letters A, B and C – called variables – which hold the numbers to be added.

We did not show the input or the print instructions in the assembly language program, but we show them below using BASIC.

```
INPUT A, B
LET C = A + B
PRINT C
```

When the three line BASIC program has been entered through the keyboard it can be executed by typing the command RUN. (This may be slightly different on some small computers). The computer will print a question mark – or display it on the screen – and the user enters a value for A, it prints a second question mark and a value is entered for B. Almost instantly, the value of C will be printed.

It should be clear by now that BASIC makes programming very much easier.

Micro Programming

It is possible on some computers to go down to an even lower level than machine language, called micro programming. Although we described the C.P.U. as being made up of three units, it is more accurately described as three groups of units. Micro programming can be used to open and close doors – metaphorically speaking – and cause a series of 1s and 0s to pass between the various units to achieve the operations required. As you might imagine, it is a fairly complicated process and is not much used. But for some purposes it does result in extremely efficient programs.

Backing Store

Even on the largest computers the main store is rarely large enough to hold all the data that the user needs to store. For this reason, magnetic tape and magnetic discs – called backing store – are used to provide extra storage capacity.

Most people will be familiar with computer tape units, as whenever a computer is shown in a T.V. play, the tape units are most prominent.

Data is stored serially on magnetic tape, which means that if data has to be read from several locations, the tape will have to be continually rewound. Consequently, the time taken to locate and read a particular piece of data – the access time – is quite long.

Magnetic discs store information randomly; that is, any piece of data can be accessed immediately, in contrast to tape which often has to be wound through most of its length to locate some item. The disc spins at high speed and the read/write head can be moved from the edge to the centre, to locate a particular piece of data, in much the same way that you can choose to play a particular track of an L.P. gramophone record. As a result, the access time for discs is much faster than for tape. But both are much slower than the computers main store.

Discs are almost never used with personal computers due to their cost but tape is, usually in the form of cassettes.

RAM and ROM

The letters RAM stand for random access memory. Random, is not used in its usual sense, but rather, it means that any location can be accessed as required. The computer main store is constructed from RAM and each location is identified by a number, referred to as its address.

ROM stands for read only memory. The interpreter of a personal computer is held in ROM because, as we said above, discs are too expensive.

When a computer is switched off the main store is emptied but ROM retains data even without the power on, so the compiler will still be there next time you want to use it.

If you have a lively mind, this article will have raised many more questions than it has answered. But by now you should have a pretty good idea of the framework of computing and thus find it considerably easier to fill in the details.

Finally, do not allow your present lack of knowledge to deter you from pursuing computing, either as a hobby or for business purposes. You will find that learning about the subject is easier and more interesting than you ever imagined.

PCW An extract from a letter the author wrote: "I have always taken the view that several hundred people with O levels are of more use to Society than one Ph.D. in a sea of illiterates".**PCW**

PCW
TROBE...STROBE...STROB

Newsletters We've been receiving samples . . newsletters such as The Nascom *MC News*, The Pet *Newsletter*, Liverpool University Computer Laboratory's *Microswop*, Southampton University's *Benchmark*. Our reaction – grassroots computing is in a terrifically healthy state. Liverpool University's *Microswop* has interesting articles such as "Microprocessors aid the blind", and the Nascom newsletter is written in the style of a letter to friends; the same goes for the others. Of course, one has to mention the ACC newsletter which has been so ably edited by Mike Lord.

IF YOU CAN'T BEAT THEM :-

VECTOR GRAPHIC INC

VECTOR V18A slot Motherchassis accepts the wide range of Vector S100 cards and makes an ideal base to build a microcomputer system. Computing power is available to perform a wide range of tasks from industrial control to small business.

£350

PR2 12K PROM/RAM card holds a comprehensive monitor program for system testing and configuration. Normal operation is in conjunction with a serial terminal via I/O card.

£160

FLASHWRITER, memory mapped VDU with graphics, allows a system to be built without a terminal or I/O card. Specify version EV of monitor program.

£150

I/O, Switchable 110 to 9600 baud serial interface plus two 8-bit parallel I/O ports.

£125

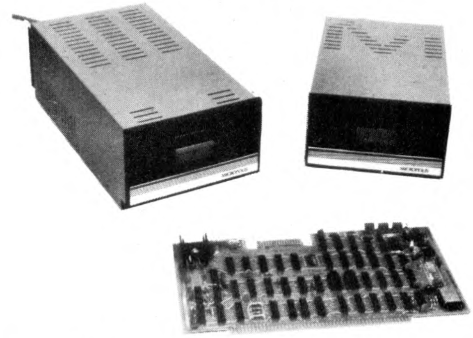
High resolution graphics interface bit-maps 8K of RAM to 256 x 256 points, or 128 x 128 with 16 level grey-scale.

£150

Z80 Processor card	£140	8080 Processor card	£120
8K Static RAM 4MHZ	£140	16K Static RAM 4MHZ	£300
Analogue Interface	£70	Precision analogue interface	£250
Rackmount 18 slot motherboard	£150	Rackmount power supply	£90

8K Computer System £895

MICROPOLIS



Micropolis disk drives employ higher standards of engineering to pack either 143K or 315K bytes per diskette, formatted, Supplied complete with controller card, cables, manual and software they plug directly into the S100 bus; 8080 or Z80.

Extended disc BASIC, mnemonic editor and assembler are provided, to run under the powerful MDOS operating system.

Add-on units are supplied to extend the system to four drives and one drive per system may be powered from the S100 bus.

143K System S100 powered	£439	143K Add-on S100 powered	£279
143K System Mains powered	£499	143K Add-on Mains powered	£339
315K System S100 powered	£649	315K Add-on S100 powered	£349
315K System Mains powered	£699	315K Add-on Mains powered	£399
Twin drive System 630K	£1159	Twin drive Add-on 630K	£859
S100 bus regulator	£14	Diskettes per five	£24

Unmounted drives available from £225

JOIN THEM !

VECTOR MZ

£2300

Combining the best features of the VECTOR GRAPHIC computer and twin MICROPOLIS 315K byte drives. The Vector MZ produces, in one package a powerhouse of Microcomputer ability.

The VECTOR PROM monitor bootstraps directly to either MDOS, for housekeeping and Assembly language operation, or to BASIC to run high-level user programs. Provision is made to immediately attach a printer, for example one of the extensive range from Centronics sold by Sintrom, enabling use of the powerful printer-related features in the MICROPOLIS Software.

Applications Software for the VECTOR MZ now in preperation will perform a wide variety of business functions; stock control, invoicing, ledger and mailing lists. Further applications exist as a microcomputer development system, and low cost replacement for minicomputer control and instrumentation.

VECTOR MZ configuration includes:

Twin disk 630K minifloppy.
Full Micropolis disk software.
Z80 4MHZ 32K processor.
1 Serial port, 2 Parallel ports.
12K PROM RAM card with extended monitor.

And will support:

Flashwriter, Graphics interface, Analogue interface
Additional RAM, additional minifloppy drives.

Centronics Microprinter	£398
Centronics 779	£780
ADM3A VDU	£620

SINTROM GROUP

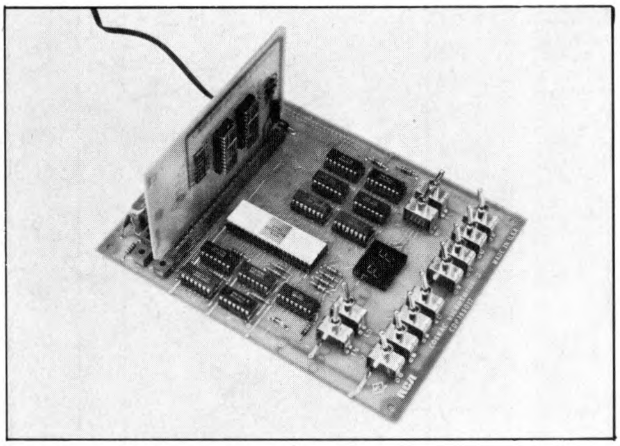
PRICES EXCLUDE VAT
ALL EQUIPMENT FULLY ASSEMBLED AND TESTED
OEM AND DEALER ENQUIRIES WELCOME

DEMONSTRATIONS IN OUR SHOWROOM

Sintrom Microshop

14, Arkwright Road,
Reading, Berks. RG2 0LS.
Tel: Reading (0734) 84322
TELEX: 847395
CABLES: SINTROM READING

A MIGHTY MICROMITE IN ACTION



T. F. Lenihan
RCA Laboratories, Princeton,
NJ, USA.
and
Les Thurlow
RCA Solid State – Europe

A single-wire alarm system using the COSMAC microtutor

A microprocessor-controlled single-wire burglar alarm system gives the user the benefits of standard multi-wire systems, but has additional flexibility and is simpler to install.

Multiwire alarm systems can easily identify the entry point in a burglar alarm system, but they are expensive and installation is complex. Conventional single-wire systems, while inexpensive and simple to install, cannot pinpoint exactly where the break-up has occurred. However, by using a series string of resistors around the perimeter that is being protected, and taking advantage of the voltage divisions present in such an arrangement, it is possible to detect the exact entry point while retaining the ease of installation inherent in single-wire systems.

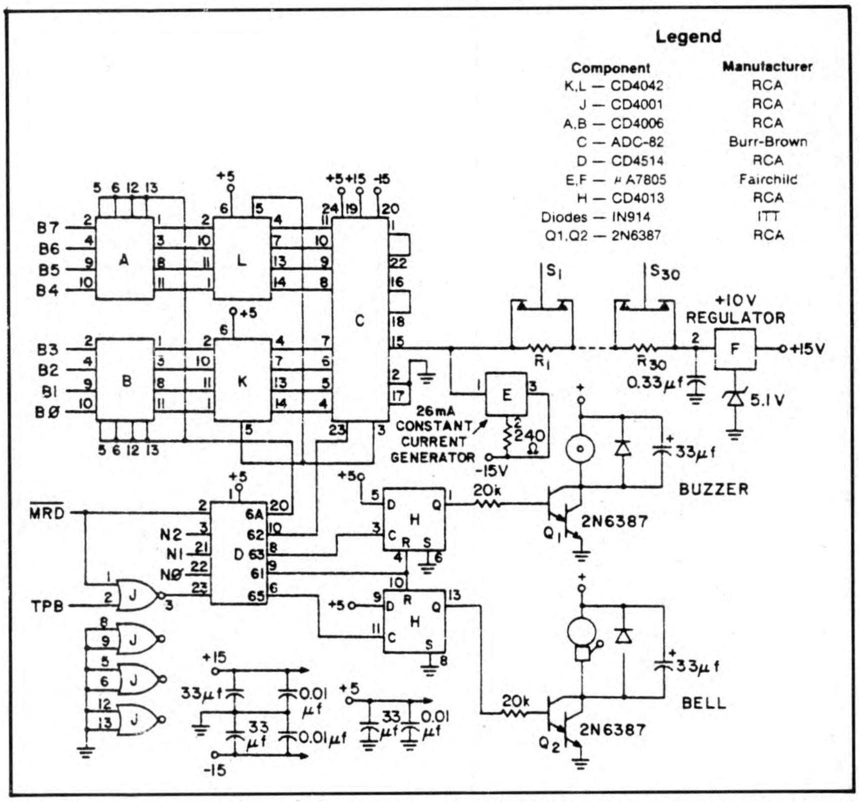


Fig. 2. *Schematic of alarm interface board shows Microtutor data bus connections B0-B7, input/output signals N0, N1, N2, and timing control signals MRD and TPB available at the external device connector of the COSMAC Microtutor. In a multiloop system, only resistor switches S1-S30, the constant-current generator and 10V regulator would need to be duplicated for each loop.*

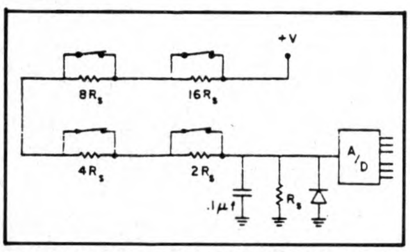


Fig. 1. *Simple resistor-switch loop system with only an analogue/digital convertor works as a single-wire system but with limited flexibility.*

Hardware

Fig. 1 shows a basic alarm circuit which combines ease of installation with the ability to locate the entry point. Although it is a single-wire system, the analogue/digital (A/D) convertor can identify the exact location where an intrusion takes place. This arrangement uses a resistor/switch combination at every door and window being protected. A different value resistor is placed across each switch in a series arrangement of switches. When a switch is activated, a unique voltage is read across the sensing resistor R_S, which is shown returned to ground.

However, this simple arrangement works for only about four entry points, i.e. four resistor-switch sensors, before the system breaks down. In practice, the string of resistors must be fed by a constant-current generator. By holding the source voltage and current to constant values, and by varying the resistance alone, the error voltage is developed across the constant-current source. Now we have a system that will cover a multitude of entry points, albeit one that lacks flexibility. For example, entry points intentionally left open cannot be ignored in this system. By adding a *microprocessor* to the A/D system, this flexibility can be achieved with software. For example, the microprocessor program could be written to disregard a window intentionally left open for ventilation, or perhaps, for a door under repair.

The circuit shown in Fig. 2 interfaces directly with a COSMAC Microtutor and will protect about 30 entry points. It consists of a 26 mA constant-current loop monitored by an 8-bit A/D convertor. The output of the A/D is latched for input to the central processing unit. The output portion consists of a warning buzzer and alarm-bell circuit. The diodes and capacitors in this section are needed for coil suppression. The circuit uses one input and four output strobes decoded by the CD4514. Another application for this circuit is for use as a keyboard. (In the simplest form of the circuit, the microprocessor could be replaced by hard-wired logic; again, however, the flexibility of the system would be lost.)

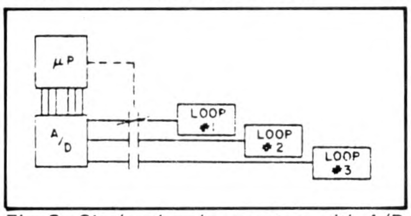


Fig. 3. *Single-wire alarm system with A/D converter plus microprocessor can be expanded easily to a multiloop system by having the microprocessor poll each loop.*

An expansion of the system organisation for protecting larger areas is shown in Fig. 3. In this multiloop scheme, the microprocessor sequentially polls several lower-resolution loops. This system will cost less, because the resistor at each entry point can have a wider tolerance, and will also have improved reliability because cutting one wire will not incapacitate the entire system.

Software

Fig. 4 is a flowchart of the COSMAC microtutor program for the burglar-alarm system. The circuit is reset immediately upon starting and, after a 20-second delay (to enable one to exit), the program reads the A/D latch. To provide a usable margin between adjacent readings, the input byte is shifted three times to the right. To protect against random noise spikes, the program requires 15 consecutive nonzero readings before it advances to the output mode. In the output mode, the entrance number is displayed in hex notation on the Microtutor and the warning buzzer is activated for 20 seconds. If the 'in' switch is not pressed within 20 seconds, the alarm bell is activated. Pressing 'in' resets the alarm and restarts the program after a 20-second delay.

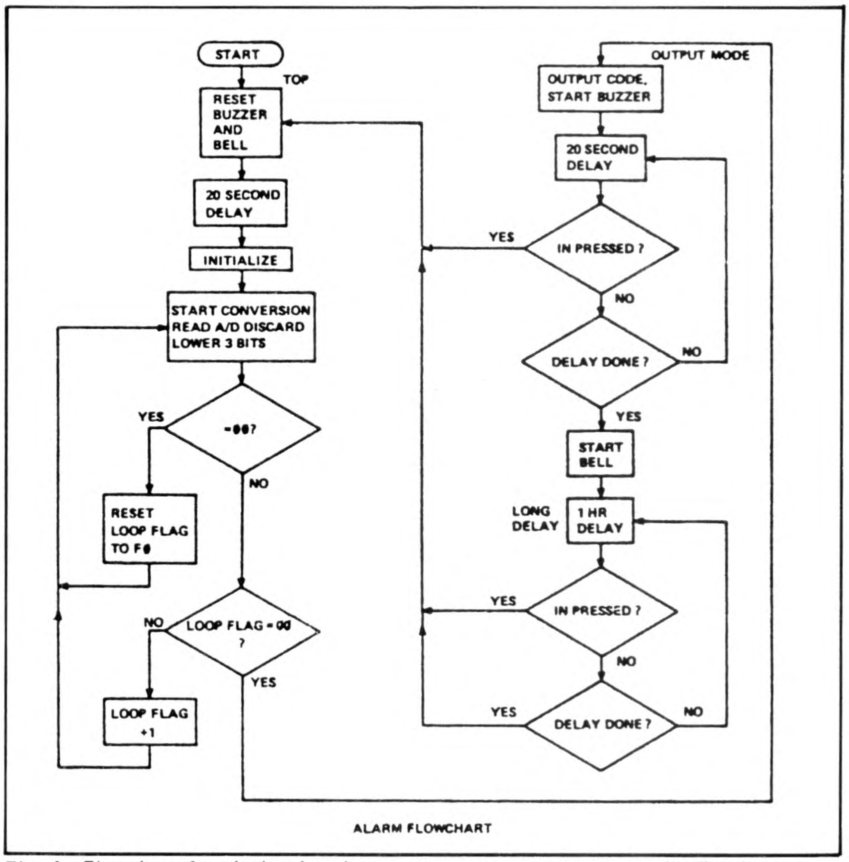


Fig. 4. *Flowchart for single-wire alarm system; program steps are detailed in Table 2.*

The program activates the warning buzzer to alert the user to reset the alarm before the bell goes off (if one is entering) or to indicate to an intruder that a circuit has been tripped before entry has been fully gained. Hopefully, this initial alarm will scare off the intruder.

The bell will reset and re-arm automatically one hour after going off, so that if the user is away for an extended period, the bell will not ring continuously until he returns. (A notice to this effect should be posted conspicuously for the police.)

Table I

Microtutor register assignments used in the alarm-system program.

R0 = Program counter
R3 = Data pointer
R4 = Loop flag
R5 = Short delay counter
R7 = Long delay counter

Delays are based on clock speed of 1.72 MHz. Change M004E to vary alarm on-time in 1-min increments, e.g.:

0A = 10 min
0F = 15 min
1E = 30 min
2D = 45 min
3C = 60 min

Table IIa

Actual alarm-system program is shown in first set of data below. Second set of data (Table IIb) ties in with flowchart (top left) and shows what event occurs at a particular instruction byte in that memory location (M).

Memory byte (M)	Instruction byte (m)							
0000	D0	E0	61	00	F8	FF	B5	A5
0008	F8	0F	FF	01	3A	0A	25	95
0010	3A	08	F8	00	B3	F8	FF	B5
0018	A5	F8	67	A3	F8	00	B4	F8
0020	F0	A4	E0	62	00	E3	6A	F0
0028	F6	F6	F6	53	32	49	84	32
0030	34	14	30	22	E0	63	00	E3
0038	64	23	F8	0F	FF	01	3A	3C
0040	25	37	01	95	3A	3A	65	30
0048	4D	64	23	30	1C	F8	3C	A7
0050	B7	F8	FF	A5	B5	F8	2F	FF
0058	01	3A	57	25	37	01	95	3A
0060	55	27	87	3A	51	30	01	

The software for implementing this program is shown in Tables I and II. The software was written to accommodate both the CDP1801 and the CDP1802 versions of the microtutor. If the CDP1801 version is used, change the 64 instructions at M(0038) and M(0049) to 60.

The code 00 is the normal closed-loop reading. Any other reading indicates a breach of the system. The unique code 1F indicates that the system wire has been cut.

Table IIb

Algorithm	M
Top	0001
Reset bell & buzzer	0002
20-second delay	0004
Initialize	0012
Start conversion	0022
Read A/D	0026
Shift right 3 times	0028
Store	002B
= 00?	002C
If yes, read again, fix pointer	002D
If no, check loop flag	002E
= 00?	002F
If yes, go to output mode	0030
If no, increment flag, read again	0031
Output mode	0034
Start warning buzzer	0035
Output code	0037
Delay 20 seconds	003A
Reset?	0041
If yes, go to top	0042
If no, continue delay	0043
Start bell	0046
Go to long delay	0047
Output code	0049
Go to read again	004B
Long delay	004D
Set up counters	004E
Change value at M004E	0050
To vary alarm time	0051
1-hour delay	0054
Decrement counter	005B
Reset?	005C
If yes, go to top	005D
If no, continue	005E
Decrement counter	0061
Done?	0063
If no, count again	0064
If yes, go to top	0065

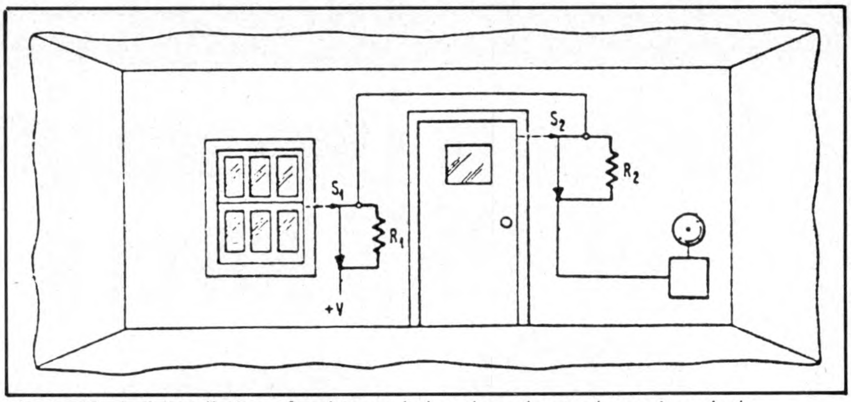


Fig. 5. *Typical installation of resistor switches throughout a house in a single-wire alarm system.*

Table III
Resistance values for each entry point to be protected in single-wire alarm system.

Resistance value (ohms)	*Entry point number (hex readout)*	*Resistance value (ohms)*	*Entry point number (hex readout)*
33	01	390	10
47	02	420	11
75	03	440	12
100	04	470	13
120	05	500	14
150	06	530	15
160	07	550	16
200	08	570	17
220	09	600	18
240	0A	630	19
270	0B	660	1A
290	0C	680	1B
310	0D	710	1C
340	0E	750	1D
370	0F	820	1E

Construction hints

The ±15V power supplies to the A/D converter should be bypassed as close to the package as possible, and the μA 7805 regulators should be installed with a small heat sink. If magnetic switches are used, they should be of the type which are normally open (contacts apart). Using the resistance values shown in Table II, the unit will monitor 30 doors and windows, a combination covering most houses. All values in Table III were determined using standard 5% resistors. Fig. 5 shows a typical installation of switches throughout a house.

The Janus Interface

A PET SOFTWARE UA(R)T

Peter Mather

Peter Mather is a computer officer at the Computer Centre, University of Birmingham, and is completing a Ph.D in Psychology there. This ingenious little article was arranged when he met the editor at the **PCW** *Show last September.*

A major problem with the PET computer has been the difficulty of obtaining hard copy. Although firms are advertising printer interfaces, the waiting lists for these units are normally unreasonably long. This program (diagram 1) was written with the intention of very simply getting round the problem of connecting a PET to a *standard* printer, such as an ASR33 TTY.

Statements 10 - 120 poke into the second cassette buffer a machine code subroutine which outputs, on bit 0 of the I/O port, the argument of the USR routine as a serial string, complete with one start and two stop bits. Statements 130 and 140 set the baud rate for transmission, and may be omitted for 110 baud.

After the I/O port has been set up and the start address of the USR routine input (Statements 150 - 180), the program reads data off the PET's cassette and sends it to the teletype, allowing 72 characters/line and appropriate line feeds and carriage returns. Upon encountering an "end of file" the program terminates (Statement 220).

This program will directly read and print data off the cassette. However, in order to get a program listing, the full source must be stored on cassette using:—

```
OPEN 1,1,1
CMD 1
LIST
CLOSE 1
```

as the "saved" program is not a standard listing.

A circuit for connection of the PET to a typical 20ma current loop TTY is given in diagram 2. It should be noted that current is flowing when the teletype is waiting for input and R1 should be adjusted to give the 20ma current in this state. A value of 500 Ω will be about right with a 12 volt supply. The motor servo is left continuously in circuit across the supply.

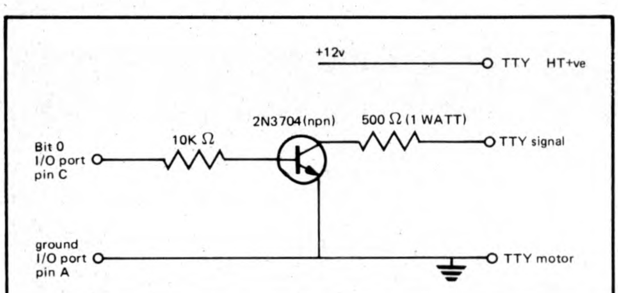


Diagram 2

```
10 DATA 32, 167, 208, 120, 165, 180, 141, 81
20 DATA 3, 32, 140, 3, 169, 0, 141, 79, 232
30 DATA 32, 140, 3, 160, 8, 78, 81, 3, 176
40 DATA 5, 169, 0, 76, 116, 3, 169, 1, 141, 79
50 DATA 232, 32, 140, 3, 136, 208, 235, 169
60 DATA 1, 141, 79, 232, 32, 140, 3, 32, 140
70 DATA 3, 88, 76, 120, 210, 162, 35, 173
80 DATA 73, 232, 205, 73, 232, 240, 251, 202
90 DATA 208, 245, 96
100 FOR I=850 TO 921
110 READ N:POKE I,N
120 NEXT I
130 INPUT "BAUD RATE"; B9
140 POKE 909, INT(3900/B9)
150 POKE 1,82
160 POKE 2,3
170 POKE 59459, 255
180 POKE 59471, 255
190 OPEN 1, 1, 0
195 K=0
200 GET # 1, M$:IF M$="" THEN 200
201 K=K+1
202 IF K>72 THEN GOSUB 300
210 PRINT M$;
220 IF ST=64 THEN CLOSE 1:END
230 T=ASC(M$)
240 H=USR(T):IF T=13 THEN 260
250 GOTO 200
260 F=USR(10):K=0:GOTO 200
300 K=0:F=USR(13):F=USR(10):RETURN
```

Diagram 1

More on the 8086

T. M. Dixon, Peterhouse, Cambridge

PCW *Readers should refer to Bill Davy's article in the September issue.* **PCW**

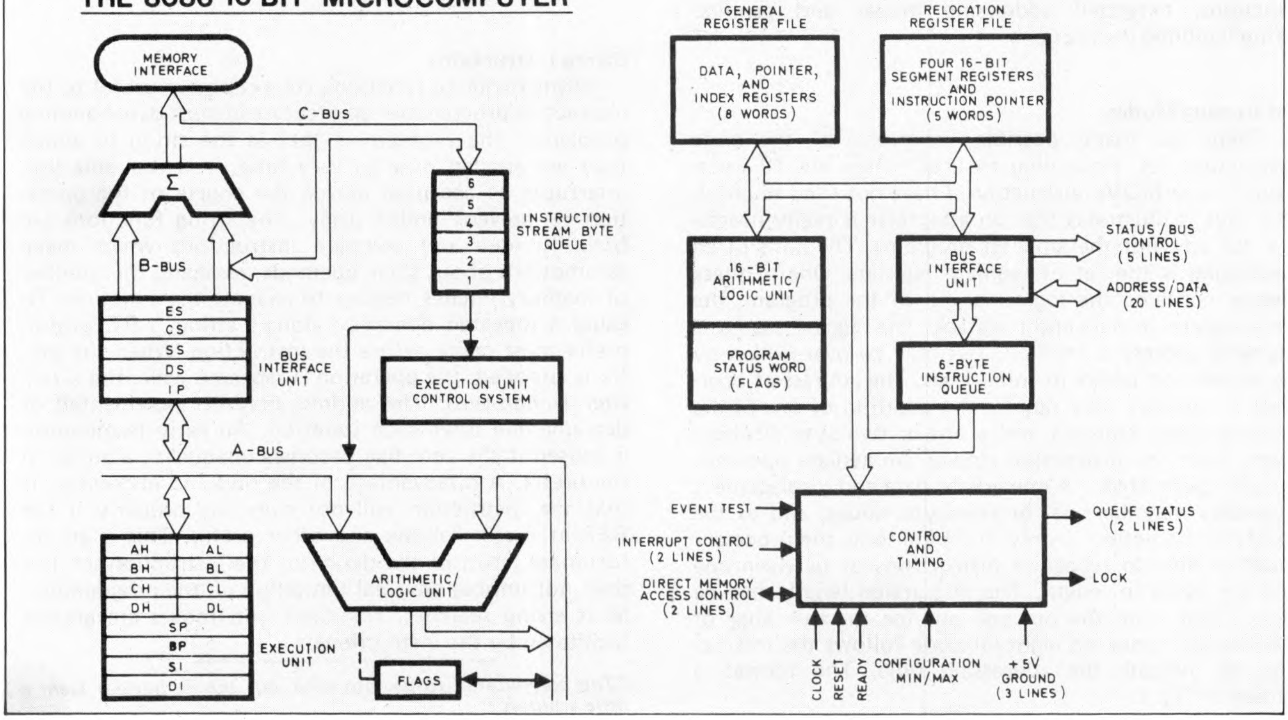
I am one of the many people who have been watching the journals keenly for news of the next generation of 16-bit microprocessors. Inspired by the enthusiasm of PCW (September 1978) for Intel's new 8086, I set out to find some further information. What follows is a quick trip around what I feel to be the more important features of the chip for assembler-level programmers.

The two biggest headaches of assembly programming for most small machines are, to my mind at least, the lack of multiply/divide facilities and the considerations necessary to produce easily relocatable code for frequently used subroutines. We have not yet reached the era of on-chip floating-point arithmetic, but the impressive instruction set of the 8086 points along this road. To my knowledge, the 8086 is also the first microcomputer CPU to use a set of registers to contain dynamic base addresses and thus permit not only much simplified and more rapid program-loading, but also dynamic relocation (try that with a SC/MP). A quick overview of the 8086 architecture will help to explain this.

The 8086 chip consists of two distinct blocks, the Bus Interface Unit (BIU) and the Execution/Control (EU). The BIU has the task of locating the next instruction for the EU and placing it in an instruction stream queue six bytes long. Keeping the queue full increases the processor throughput as memory fetches can occur while the previous instruction is being completed. To access the 1 MB memory, the BIU takes the IP register (Instruction Pointer) which is 16 bits long, and adds one of the segment registers (extended to 20 bits with 4 low-order zeroes) to produce a 20-bit address. To access operands, the Effective Address (EA) so produced may optionally include a base and/or index address taken from the registers of the EU, and a displacement.

The EU holds the general registers (A, B, C, D) which are each 16-bits long. These may be used as word-registers or as eight-byte registers. The A-register is in some cases used as an accumulator, and the other registers have dedicated uses. All registers can be used interchangeably for many instructions. In addition, four 16-

THE 8086 16-BIT MICROCOMPUTER



bit registers are available as 'Pointer and Index' registers. These may be used to permit based and indexed addresses; one register is a dedicated Stack Pointer. The EU also holds a set of flags indicating the processor status. The register sets are more or less logical extensions of the 8080 registers.

7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	15 ... 7 ... 0	15 ... 7 ... 0
op-code	address type	immediate data	address displacement
1 byte	1 byte	1 or 2 bytes optional	1 or 2 bytes optional

The address type code has the following fields:–

7 6	5 4 3	2 1 0
mod	reg	r/m

If mod = 11 then there is no memory access, and r/m indicates the second register operand.

Otherwise mod indicates the presence and length of the displacement.

The reg field indicates the register operand (coded according to instruction type).

The r/m field indicates the index/base registers to be used in addressing the memory operand. Up to 2 such registers may be chosen from a pool of 4.

The presence and length of immediate data is indicated by the op-code.

Fig. 1. 8086 Instruction Format (usual).

The instruction set of the 8086 is also an extension of that of the 8080. All 8080 instructions have an equivalent operation on the 8086, but the mnemonics for the instructions are frequently different and the object-code is not compatible. It is thus necessary to re-assemble 8080 code to run on the 8086; and, to obtain the claimed order of magnitude performance increase, it will probably be necessary to rewrite at least the more critical code. The standard clock speed of 5MHz is probably faster than most 8080 systems.

The more or less standard set of load/store/push/pop instructions is provided, with of course far more addressing modes, and the same is true of the logical and call/jump/return instructions. I/O instructions are allowed to fixed or variable ports (i.e. immediate or register-contained port address), and memory-mapped I/O can be used should these facilities not be adequate. The advanced features are the extended set of arithmetic functions, extended addressing modes and genuine string-handling instructions.

Addressing Modes

There are many possible variations of the basic instruction set. According to Intel, there are 19 variations of the MOVe instruction. I have not tried to check this, but it illustrates that an assembler is really necessary for any but the smallest programs. The basis of all addressing is the set of segment registers. One of these always contains the logical origin of the program, the others point to data and the stack. The register used as a segment address is implicit, but may be over-ridden by an instruction prefix in some cases. The address of operands in memory may require the addition of one of the base or index registers and a one-or two-byte displacement from the instruction stream. Immediate operands are also permitted. As immediate data and displacement addresses may be one- or two-byte values, and as the shortest instruction is only one byte long, the processor must be able to recognise instructions of between one and six bytes in length. The instruction length may be determined from the op-code but for the full range of addressing modes, an eight-bit code follows the instruction to indicate the addressing type. The format is shown in Fig. 1.

The organisation of the instruction set appears to be the greatest weakness of the 8086. Presumably in order to reduce the redundancy of information in the instruction, many instructions which do not require the explicit provision of a full address use shortened forms. The instruction to POP a register from the stack is shown in Fig. 2. Whilst this form of instruction is economical on storage, it does eat up eight possible op-codes, one for each register. Most op-codes contain a flag to indicate the length (byte or word) of the operands and possibly further flags. The set of op-codes thus runs out quite quickly, with strange results. Signed and unsigned multiply and divide instructions all have the same op-code and are distinguished only by a sub-code in the register field of the second byte of the instruction. This effectively precludes the use of registers other than the accumulator as the target of these instructions and would seem to prevent the design of upward compatible chips with extended features in the future. The rules which govern the permissible combination of registers to produce an address seem over-complicated,* and the availability of two-levels of indexing seems over generous. I do not like the use of instruction prefixes for any task and in particular would like to see a segment register field in the instruction or some other addressing scheme which perhaps could treat all registers as equal and use only one level of indexing, but perhaps I am too used to IBM 370 machines.

Arithmetic Instructions

Despite my comments about the multiply/divide instructions, the facility is very well worth having and apart from the register limitations, is quite comprehensive. Half- and full-word binary quantities may be multiplied with or without sign to give respectively word and double-word results. Instructions are available to extend the sign - bit of half- and full-word operands to double their length. The divide instruction operates on a double length quantity to produce a single length (8- or 16-bit) quotient and a single length remainder. Adjustment instructions allow division of unpacked (single byte) decimal quantities giving an unpacked decimal result. Corresponding facilities allow unpacked decimal numbers to be added, subtracted and multiplied. Packed decimal operands may be converted for the addition and subtraction instructions.

String Instructions

String-handling functions are extremely useful to the interactive programmer and feature in most data-handling problems. The problem is that as the string to which they are applied may be very long, it is desirable that interrupts be accepted during the course of the operation to prevent undue delay. The string functions are basically load and compare instructions which make assumptions about their operands, reducing the number of memory-fetches needed to execute the function. To cause a repeated operation along a string, a REPetition prefix must come before the instruction. When the prefix is supplied, the operation is repeated while the C-register is non-zero. The address may be incremented or decremented after each iteration. An early termination is caused if the zero-flag becomes unequal to a bit set in the prefix. A disadvantage of the prefixed instruction is that the instruction will not complete properly if the REPeat prefix follows any other prefix. This is an unfortunate result of the design of the instruction set, but does not impose any real limitation on the programmer. Most string searches, translates and moves are greatly facilitated by the instructions.

**Too complicated to go into here, but the limitations seem a little arbitrary.*

Control Instructions

A number of control instructions are provided to facilitate the use of many devices on one bus system. The WAIT instruction causes the processor to wait for an external signal on pin 23 before continuing; the LOCK instruction (another of those prefixes) causes a signal to appear indicating that the processor is demanding control of the bus. The most interesting instruction is ESCape. This is effectively a no-op, but places the effective address (computed using the standard 8086 addressing modes) of the operand on the data bus to allow other processors to make use of the addressing features of the 8086 and access data in relocatable storage. Various control signals which are also available, externally, may be used depending on the system configuration.

7	6	5	4	3	2	1	0	
0	1	0	1	1	reg			POP instruction with register operand.

7	6	5	4	3	2	1	0	7 6	5 4 3	2 1 0	
1	1	1	1	0	1	1	x	mod	code	r/m	Multiply and Divide instructions.

(x indicates operand length)

Fig. 2. Comparison of POP and Multiply/Divide Instructions.

Peripherals

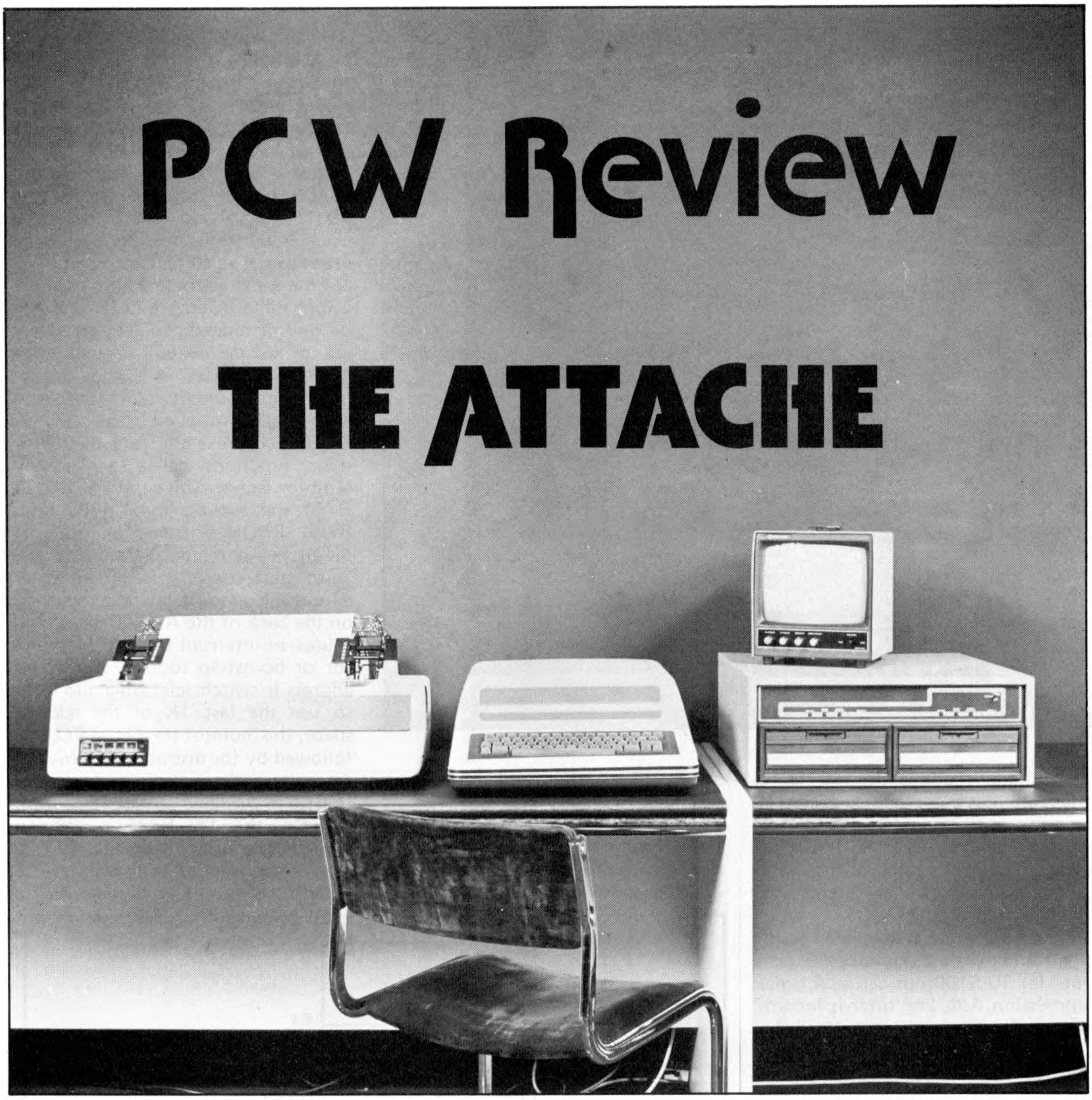
Two devices which are not strictly peripherals are the 8284 clock generator, which can be used with a crystal to provide the necessary clock signals and generate a suitable RESET signal, and the 8288 Bus Controller. The 8288 is used only in larger systems to fully decode the control lines. The 8259A Interrupt Controller sorts out priorities and interfaces to the 8086 vectored interrupt system. Some new peripheral chips which are not specifically designed for the 8086 but which will increase its applications are the 8271 Floppy Disk Controller, the 8273 HDLC/SDLC controller which will support the newer line protocols and the 8291 interface to the IEEE488 Bus. Other planned devices range from dot-matrix printer controllers to encryption chips.

The sad fact is that these many facilities are only likely to be in the reach of most of us if the price of the 8086 and its support devices drops quite considerably. There are two Intel-developed kits which could conceivably be within the reach of the hobbyist, a Component Evaluation Kit, and a System Design Kit (SDK-86). The former, I am informed, consists only of the integrated circuits necessary to develop a minimum system, and costs about £250. The latter is a hex-keyboard/seven segment display kit with a simple monitor, 2K memory and TTY support. I have been quoted prices ranging from £400 to £700, but no-one was very confident about their estimate. On the face of it there would seem little to justify the high cost when one considers that the 8085 System Design Kit (SDK-85) costs only about £160. Intel claim that the small die-size of the 8086 chip will lead to reduced cost as production experience (and presumably demand) grows. When one considers the amount of hardware and software needed to make really effective use of such a powerful tool, it would seem that the 8086 is still a device of the future for the hobbyist.

Note

Most of the information in this article is drawn from the Intel MCS–86 Preliminary User's Manual. Both **Intel** and **Rapid Recall Ltd.** were both of great help in obtaining information. Information taken from the MCS–86 manual may be subject to change and may be protected by copyright.

PCW Next issue: The Motorola 6809 MPU.



The Attache

Martin Healey
University College, Cardiff

1. INTRODUCTION

It is a true reflection of the maturing state of the microcomputer industry that we have a new product line from an existing manufacturer. In fact the ATTACHE is the latest product from the original personal computer company, the makers of the ALTAIR, now under the wing of leading U.S. peripheral manufacturer PERTEC. PERTEC Computer Corporation Microsystems Division markets under two product names in the U.S.A., MITS and iCOM, the Attache appearing to come from the latter stable. MITS are well known in the U.K. and I have used an iCOM floppy disc system on an INTELLEC MDS with no problems for some years now.

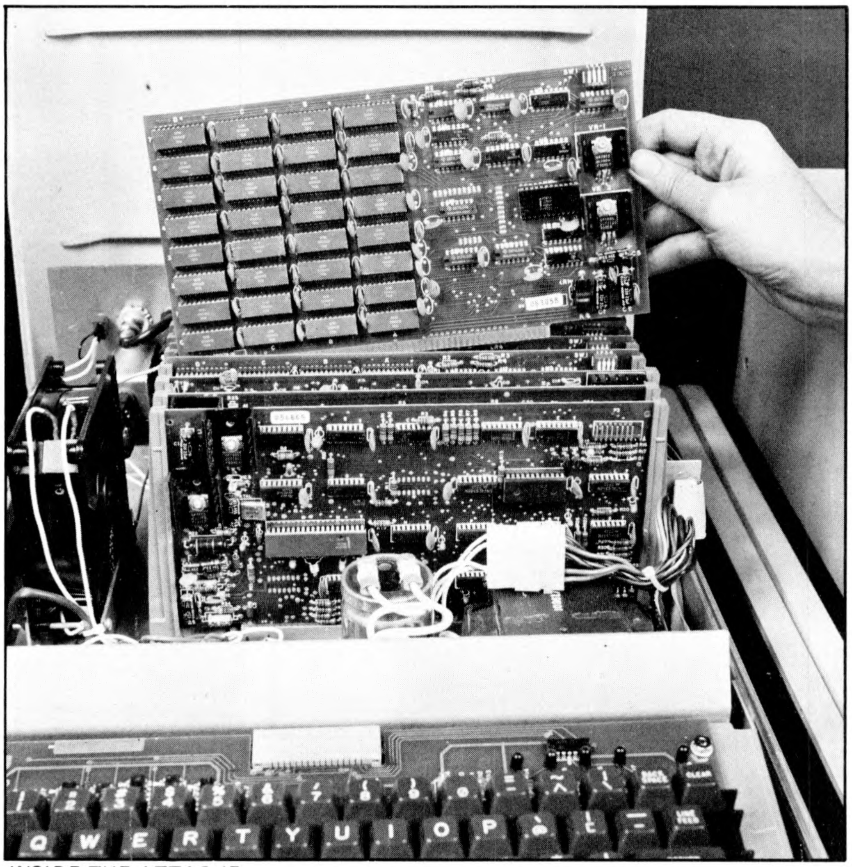
In practice the ATTACHE is a repackaging of the ALTAIR components, utilising quite naturally the S100 bus structure. Thus, while there are some new cards, many well tried and tested components are incorporated.

The major new feature of the hardware is in the packaging, which follows the concept of the Apple by embodying the keyboard in the computer "box" with a video output socket to drive a stand alone monitor. The ATTACHE however is not aimed at the low cost hobbyist market. Bulk data stores are incorporated in separate stand-alone boxes and although a cassette tape interface is available, the ATTACHE is really intended as a floppy disc system. As such it uses MITS BASIC, with disc handling, as its normal user interface.

Unlike the ALTAIR, the ATTACHE is marketed by MONCOLAND, the leading light of which is Derek Moon. This is a further reflection of growing maturity and commercial interests since he has his roots in commerce and retail marketing and not in computing or amateur enthusiasts. Thus while the ATTACHE has some appeal as a personal computer it is as a small business computer that it will really be marketed. MONCOLAND are actively encouraging marketing through standard (electrical) retail outlets; they are totally committed to providing applications software packages and organised maintenance.

2. HARDWARE

The ATTACHE is, as only to be expected, a pleasantly packaged machine. Good simple aesthetic appeal is of course important if the machine is to sell to business users. The in-built keyboard is a 64 Key unit with a good "feel" and a conventional lay-



INSIDE THE ATTACHE

out (thank goodness!). It is a "stepped" keyboard and as such will be quite acceptable to an unskilled operator, helped by the QWERTY layout. The importance of a quality keyboard in commercial systems cannot be overstressed.

The box contains the power supply and a fan with a motherboard with slots for 10 S100 bus cards. A noise suppression A.C. line filter is incorporated. The system I tested had the usual problem child of a 220 volt transformer, but I am assured that all units are being shipped with proper 50Hz, 240 volt transformers. In all fairness the machine ran quite cool even with the 220 volt transformer. Being an S100 bus machine the power supplies provided only smoothed D.C. (+8,+18, and −18 volt), employing voltage regulators as required on each board. I personally feel that this technique has avoided a lot of potential problems with instability on all S100 bus systems. However it also excludes the use of switching power supplies, a feature of the APPLE which results in reduced weight and heat dissipation.

The CPU card is the standard ALTAIR card featuring a 2 MHz 8080A, with an 8224 clock generator but standard TTL logic for system control, latches and drivers rather than the 8228 (was the 8228 available when this board was designed?). It is surprising now to see a whole board dedicated to a CPU but all MITS boards use low density chip packing with the resulting minimisation of faults and debugging problems.

Both a static and a dynamic 16Kb RAM board are offered. The static board is quoted with an access time of 215 nanoseconds and the dynamic (or synchronous as the literature calls it) 350 nanosecond. The latter relies on timing signals from the CPU and therefore presumably inserts wait states for memory refresh cycles. The system tested had only static boards and so I was unable to make performance comparisons. Both would appear plenty fast enough for the 2MHz 8080A, in which case the cheaper dynamic memory would be attractive. The static boards supplied are the same as those I have used for a long time in an ALTAIR and can be well recommended. The addresses are of course switch selectable on 16Kb boundaries. 4Kb boards are mentioned in the literature, but these are not being supplied in the U.K.

The system employs a board for basic functions called a Turnkey Monitor Board. This supports 1Kb of RAM and sockets for 4 1702 (256 Byte) PROM's, used for optional monitors and bootstrap loaders. Auto start circuitry is initialised by depressing a toggle switch mounted on the back of the ATTACHE, which causes an interrupt to start the monitor or bootstrap routine. The ROM address is switch selectable and is set to use the last 1K of the address space, the monitor starting at $FC00_H$, followed by the disc bootstrap loader. An alternative multi-boot loader (MBL) is available for booting from cassette. the 1K RAM is situated immediately prior to the ROM, (Figure 1) and is used as a stack by the monitor. It is not at all clear how a

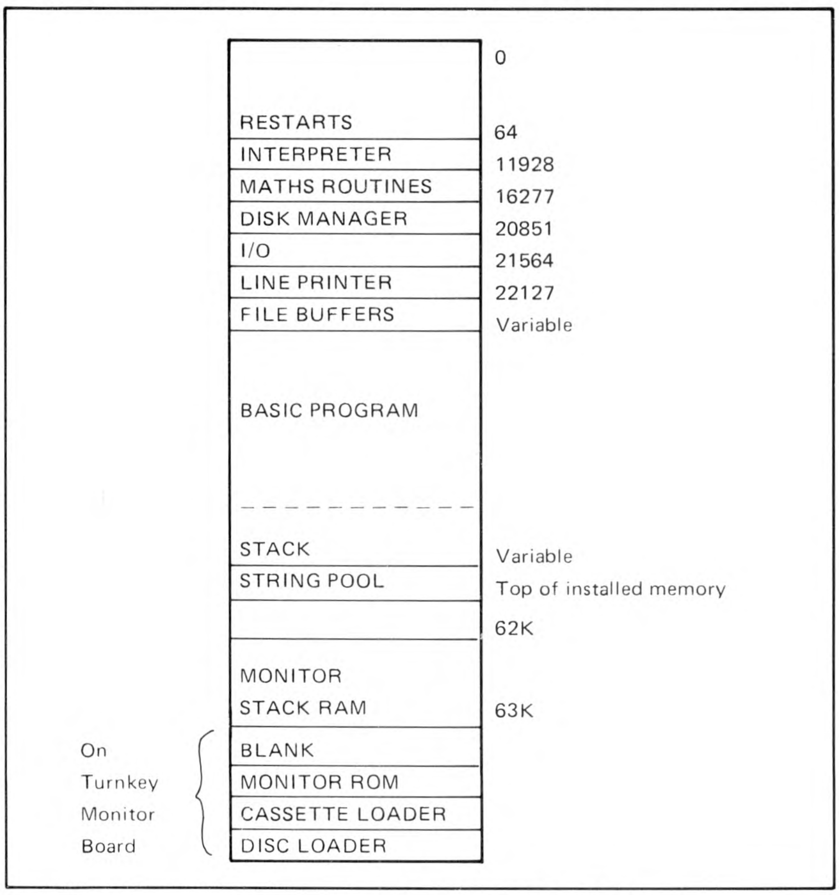


Figure 1: Memory map for the Attache with BASIC

clash of physical address space is avoided when a 16Kb RAM card is selected to the top quarter of the address space; presumably there is logic on the monitor board to gain priority for the top 2Kb. The circuits are provided but time has not yet been found to decypher these.

Also included on the Turnkey board is a UART for a serial I/O port, using ports 16 and 17. An RS232 (V24) standard outlet is provided on the front edge of the card. The board is factory set to 9600 Baud; but jumpers are provided for other speeds.

The I/O from the Turnkey board is directly coupled by a short cable to the Video board. The Video board is in fact a double board (piggy-back), screwed together with only one S100 connector. Since it is thus effectively double width it is plugged into a socket at one end of the motherboard to avoid covering up another socket. The video board has a UART for connecting to the monitor board plus a parallel port for the output from the keyboard. The actual keyboard electronics are mounted on a separate PCB under the key pad, and generate ASCII code. Thus keyboard output is routed in parallel to the Video board, in serial to the Turnkey board and thence is parallel to the S100 bus, a rather round about route. The Video board also supports a 1Kb RAM which is used to generate composite video signals for a 64 character x 16 line display. This RAM is loaded by the control logic for interpreting carriage return, etc., is included on the board together with the video refresh logic. The board tested had been modified to 50 Hz standards, generating composite video signals via coax sockets on the rear panel of the box, suitable for a monitor. An external modulator would be required to use a TV set but the lower quality is not suitable for commercial applications anyway.

The combination of Turnkey and Video boards is rather surprising as the short 9600 baud link could have been faster implemented by a parallel link. Alternatively the I/O port could have been directly serviced on the Video board from the S100 bus. As it is the Video board only picks up power from the computer bus, further there should be no need for a double board if some of the newer components were employed. On the credit side, however, the board generates a good quality display, provided a proper video monitor is employed. 64 x 16 is rather limiting for commercial applications and even though upper and lower case are supported there are no cursor controls. As already stressed the keyboard is of good standard; it also supports 5 LED's to indicate system status.

A parallel I/O board is supported, specifically for interfacing a Centronics printer or equivalent. This board is directly supported by the BASIC LPRINT statement.

Mass storage is provided by a choice of either cassette tape or floppy disc. Since this is a PERTEC product there can be little doubt that a hard disc will soon be added to the system. The cassette system is of little interest with the commercial orientation of this machine but for the record the standard ALTAIR 88-UIO single board Kansas City (300 baud) interface is available, with appropriate version of the Bootstrap Loader PROM for the Turnkey board. The floppy disc system is far more interesting. It comes in a separate metal box, attractively finished to match the processor. It houses two 8" PERTEC drives with separate status indicator lights and its own (240 volt!) power supply. The disc controller comprises two S100 bus boards utilising TTL logic rather than single chip controllers which are virtually (probably exactly) the units supplied with the older ALTAIR. The disc drive (and the printer for that matter) are connected by multicore flat ribbon cables to in line sockets at the rear of the ATTACHE.

The floppy disc drives currently supplied are standard 8" single sided, single density PERTEC FD512 with 360 RPM, 400 millisec average access time and a data transfer rate of approximately 32KByte/sec. The discettes are hard sectored, 32 sectors/track, 77 tracks; note that this is not the standard soft sectored IBM format. Using 32 rather than 26 sectors per track gives a total capacity per discette of 310 Kb, 620 Kb for the dual drive system. A dual density version is promised soon.

A number of other ALTAIR cards are offered, e.g. serial I/O, process control I/O, etc. However for a full commercial system with CPU, Video, Turnkey, 4 x 16Kb RAMs, printer and 2 x disc controller, all 10 slots are used.

3. SYSTEM SOFTWARE

The ATTACHE is supplied with iCOM software, largely based around the MITS BASIC. In fact three versions of BASIC are available, a simple 8K version, a 16K version which is available on a single ROM board as an option, and the disc extended version, referred to as MITS300-5A. This is the only one of interest for commercial programming. Figure 2 is a summary of characteristics. The system I tested used the Version 5 BASIC which must be the best BASIC available on microcomputers for commercial work. This version of BASIC has strong overtones of DEC's BASIC-PLUS, the Rolls Royce of BASIC (written by the way by MICROSOFT Corp. and available under CP/M and other operating systems on other microcomputers) and BASIC-PLUS would make interesting reading – suffice it to say that the standard is more than we dare have hoped for in such a short span of microcomputer development. MITS BASIC includes sequential and random access to disc files. An ISAM package is also available which gives access to files by key names, an im-

FEATURES	BASIC VERSION 8K	BASIC VERSION EXTENDED	BASIC VERSION MITS300–5A
Minimum Memory Requirement	8K	16K	24K
Numeric Types			
Single Precision	Y	Y	Y
Double Precision	N	Y	Y
Integer	N	Y	Y
Strings	Y	Y	Y
PEEK and POKE	Y	Y	Y
INP and OUT	Y	Y	Y
Arrays – any size or dimensionality	Y	Y	Y
IF...THEN...ELSE	N	Y	Y
PRINT USING for formatted output	N	Y	Y
EDIT command	N	Y	Y
Automatic line numbering	N	Y	Y
Error trapping	N	Y	Y
Trace	N	Y	Y
Disc files for programs and data	N	N	Y
Functions			
Intrinsic	11	22	23
User-defined	Y*	Y	Y
Machine language subroutines	Y	Y	Y

*In 8K BASIC, functions must be defined on one line and may have only one argument.

Figure 2: Summary of characteristics of the three versions of BASIC offered. MITS300–5A is, in fact,M–BASIC.

2

portant feature for a commercial system.

The system I tested was fitted with a Bootstrap ROM so that toggling the Reset button automatically loaded BASIC. There is a short dialogue to identify memory size, number of disc drives and file limits and printer type and you are away. For commercial applications a new loader is being developed to give instant access to a MENU for program selection.

The BASIC incorporates a useful edit facility. The PRINT USING and ERROR TRAPPING features are most desirable in commercial applications. Machine code functions can also be supported but must be loaded by using the POKE command, after translating binary code to decimal. There is no direct Assembler support so that this feature is of little interest here.

The literature describes an alternative discette-based software system to the BASIC supplied called FDOS-III. This is also an iCOM product of unknown origin, but it looks like a useful Assembler language program development facility.

From the literature, remembering that I haven't had the opportunity to try it yet, FDOSIII utilises a memory resident system monitor, which I presume is booted down by the DBL PROM on start-up, rather than the simple ROM monitor which is bypassed by the bootstrap anyway. The system includes a text editor with a paging command to allow page at a time in memory so that large programs can be edited – remember that the source code for a 10Kb object program will exceed 64Kb. The Assembler generates relocatable code so that a linking loader is also provided to resolve global symbol references and to create complete programs from disc resident modules. There is no mention of Macros or conditional assembly control. There is also a Debug module which allows insertion of break-points and display of register contents, etc.

The BASIC system includes a few utility programs, in particular a file transfer package called PIP. This is of course essential for making archive copies of data files in commercial applications.

4. RETAIL PRICES

The retail price list provided by MONCOLAND follows. Note that the list as published does not cover the options such as process control interfaces. Nor does it include video monitors or Centronic printers. There is no price either for the FDOSIII software or the 8K BASIC. All this firmly reflect MONCOLAND's policy of stressing commercial data processing systems.

Basic System Case + CPU + Turnkey monitor board + 16K RAM	£1,466
Video board	£ 271
16Kb Static RAM	£ 347
Printer interface card	£ 208
Floppy disc sub system 2 x 310 Kb discettes + controller + box, etc.	£1,701
Basic interpreter (on floppy disc)	£ 41

Also offered are the ROM BASIC on a single 16Kb board at £261 and the Kansas City standard cassette tape interface at £251. All prices are exclusive of VAT.

From this list a simple personal computer figuration with 16Kb and cassette interface would cost £2,000, guessing at a small allowance for 8K BASIC, excluding video monitor and cassette recorder. A full business computer system with 64Kb, twin discettes and printer interface comes out at £4,728, to which must be added a good quality video monitor at about £150 and a Centronics printer, costing around £1,400, is essential for commercial work, else preprinted stationary cannot be used. A working system will therefore cost about £6,300.

5. MAINTENANCE

In keeping with the commercial orientation of the ATTACHE, MONCOLAND have arranged a maintenance agreement with Computer Field Maintenance (CFM) for business users. The price for maintenance is broken down into units but comes out to £522 per annum for a full system plus printer and video monitor maintenance at about £350 per annum. This is a reasonable charge from a reputable nationwide independent maintenance organisation which compares well with the rates for VRC's and other small business computers. Maintenance for the business man is of extreme importance of course; he cannot rely on amateur "fix it".

6. APPLICATIONS SOFTWARE

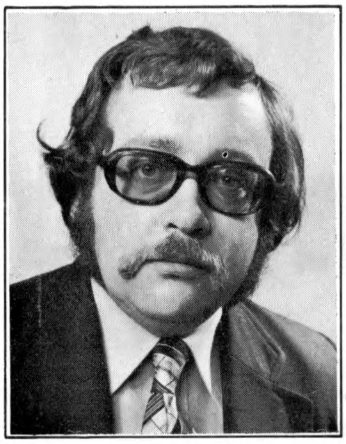
This is a real sign of the times! Already the U.S. predictions suggest that the importance of microcomputers in commercial applications will outweigh the home market. Here we have proof of the pudding in a U.K. company. MONCOLAND have invested real money in developing applications programs which are to be marketed at the quite amazing price of between £30 - £40 a module. I was provided with a pre-release of the Order Processing System which comprised four modules, Order Entry, Stock Control, Customer File Maintenance, and Invoicing. At a total cost of around £160 this represents remarkable value for money. The draft documentation looks properly organised to suit the non-computer specialist business man. The full system is due for release by mid-December. Other suites for Sales Ledger, Purchase Ledger, Nominal Ledger and VAT and Payroll are under development and will be progressively released during the early part of 1979.

I am personally convinced that provided applications programs are designed on the 80/20 principle, then by utilising potential bulk sales, software can be successfully marketed at low prices – the "Woolworth's principle". The 80/20 principle implies that the programs are specified to do 80% of the job, which is common to most users. The 20% extra will cost considerably more to provide than the basic program. I believe that for any realistic small business it makes sense to cover the bulk of the accountancy requirements at minimal cost rather than get too involved with complex computing.

7. CONCLUSIONS

The outstanding feature of the ATTACHE is the high quality of the product. It is not really a product for the home enthusiast, but a properly configured small computer. The standard of construction of the boards, and the keyboard in particular, leave nothing to be desired compared to most accounting computers. It would probably be more attractive if the screen was built into the processor/keyboard box but a small low price screen like the PET would not be acceptable for commercial applications; hence a high quality video monitor is a good compromise. The current video board only generates a 64 x 16 display; an 80 x 24 is desirable although a standard VDU could be used as an alternative to the video board – at a cost.

Frankly, as a personal computer "toy" the ATTACHE doesn't rank alongside the APPLE. The latter offers colour graphic support with BASIC programming commands and switching-mode power supplies at a lower cost, admittedly employing the slower 6502 CPU. However with a full 10 slot S100 bus motherboard, the ATTACHE is much more versatile. Fully figured with large memories and disc stores, plus the high performance BASIC, well suited to commercial applications programming, backed by details like ISAM packages it is a very good product indeed. I shall be very surprised if MONCOLAND with their applications programs and organised commercial standard maintenance don't succeed in taking a big share of the growing small business system market. Throw your Visible Record Computer (VRC) away as soon as possible and look at these types of machines – the time has arrived!



ON THE LINE

David Hebditch

2. Telecommunications Futures

In the first article in this series, I wrote about the symbiosis of computers, communications and broadcasting leading to telecommuting and Marshal McLuhan's far-sighted concept of the Global Village. The emphasis on the symbiosis is important: it is the combination of advanced computer and communications technology which is the key.

Unquestionably, the pace of computer development is much faster than that of telecommunications. There are many reasons for this but two of the major ones are that (1) telecommunications networks are much more complex and widespread than the average computer system and (2) the telephone network is dealing with analogue input and output. I shall come back to these points later.

The British telephone system is, on the whole, crude, inefficient and unreliable. The switching of lines uses the Strowger System which is an electromechanical arrangement. The basic principles of this were first devised over 100 years ago by an undertaker (the original Mr. Strowger) in the American Mid-West, who was losing business because his competitor had 'nobbled' the operator of the small-town's manual exchange. Modern exchanges use computer-controlled switching and it is no coincidence that one of Europe's most successful private branch exchanges is made by IBM (the 3750).

The Post Office has no computer-based switches installed. In the USA, AT & T have been commissioning them for well over ten years. The Post Office project to develop a new standard exchange for the future; it is called System X and looks very exciting on paper. Unfortunately, the project seems to be fraught with technical, managerial and political difficulties.

Switching is just one aspect of telecommunications; transmission technology is another. As I mentioned before, the fundamentally analogue nature of speech makes it difficult for the Post Office (or any telecommunications administration) to benefit from recent advances in microelectronics as the computer industry has. In simple terms, speech is transmitted as an electrical analogy of the sound waves made by the voice.

These signals need to be periodically reamplified; a process which does nothing to improve the signal-to-noise ratio. If speech could first be digitized then regenerative repeaters could be used instead of the amplifiers. This has two benefits; firstly the noise problem is significantly eased and, secondly, they can be made from LSI digital components. (This problem of handling analogue signals is illustrated by the fact that a standard pocket calculator can have as many as ten times more components than a colour television set).

The Post Office have been at the forefront of developments in the area of speech digitization, notably with Pulse-Code Modulation (PCM). But the performance of any telecommunications system is measured by the quality of the service received and the size of your telephone bill. Whereas advances in electronics can be readily and speedily implemented in small, autonomous units such as personal computers, the very widespread and complex structure of the telephone system makes such a change in technology a massive and costly undertaking. Almost everything has to be changed before the subscriber sees much benefit and no-one should expect to see any massive improvements before 1990.

What about the much-vaunted age of the communications satellite? As far as Europe is concerned the problem of managing the scarce resource of the available frequency bandwidth will probably restrict their use to long-haul international links (e.g. Italy-Scandanavia, UK – Greece). The future more probably lies in the use of very high capacity cables of the co-axial, waveguide and/or optical types.

Once higher transmission capacities and improved switching techniques do become available, then a whole new range of services may be possible. Getting these new services, especially electronic mail, off the ground is going to be difficult within the present Post Office structure. I think the only way it can work is if the telecommunications departments are split off to form a separate corporation.

The present management/union infrastructure at the Post Office is just not geared up to handle major technical advances. Look what happened to a very minor technical advance; the Post Code. For years the Post Office has been exhorting us to use the Post Code but hardly does so itself. The number of letters which are routed source-to-destination by code only is very small indeed.

If the Post Office in its present form cannot handle a *relatively* simple thing like the post code, there is no chance for electronic mail. Clearly, major changes are needed.

For more on telecommunications read James Martin's book 'Telecommunications and the Future'. (Prentice Hall).

Next Month: *How PCW Readers can tune their own computers to the telephone network.*

2

From Alpha to Omega
WORD PROCESSING

Charles Sweeten

There are now many offices and small business that are considering some form of word processing. Prices for the usual systems range from 3500 pounds to 65,000 pounds, with a worthwhile system for the small user, with one station and disk backing, working out at about 6000 pounds minimum. *IBM* offer a system at around 12,000 pounds which attracts many people, and the *Rank Xerox* system comes out at about 7500 pounds.

These are high prices to pay, and in addition, these same concerns may well wish to have some form of computer accounting or stock control. I want to describe a word processing system costing around 4000 pounds which will also provide computing power for the smaller office or company.

The South West Technical Products (SWTP) computer has now established itself as a useful computing tool in the business field I believe, but it seems to be less well known as a Word Processor. Indeed it is surprising that such a company as Computer Workshop who market SWTP in this country have not had a greater response from the business community to this feature of the SWTP system. Let me say that I have no connection with this company, and that I have used their system along with several others from other manufacturers, and that I look forward to the day when they *all* have Word Processing.

What is a Word Processor?

Quite simply, a Word Processing machine is one that separates the process of typing from that of printing. What is typed is displayed, and can be corrected, modified, moved around, and recorded in machine readable form. The final version or versions may then be typed out in a wide variety of ways which are governed by a set of Processing commands. Such a machine would in other circumstances be called a computer with a program in it. SWTP sell a Text Processing System which is just such a program. Naturally it only runs on their computer. The process of entering words (Text) and manipulating them is carried out by a *Text Editor.* These two programs are called off the system disk as required, and operate on any file that is held on a working disk.

However, before describing the operation of this system in detail, let me give you some facts.

The average typist spends one third of her typing time on correcting errors.

She spends a further quarter of her time in making author's corrections.

A typist in a conveyancing office may spend 90 per cent of her time in typing standard clauses.

A golfball machine types at 30 characters per second.

A daisywheel printer prints at 50 characters per second.

A dot matrix printer can print at 200 characters per second.

A keyboard and screen display can cost as little as 500 pounds and as much as 2000 pounds.

> **THE EQUIPMENT**
>
> In order to do Word Processing on the SWTP system you will need:
>
> 1 MP–68.2 Computer System; 2 MP–8 8K Memory Boards; 1 MP–L Parallel Interface.
>
> **Options:** MP–8 more 8K Memory Boards; 1 CT–64 VDU Terminal or any other VDU such as SOROC, LYME, HAZELTINE; 1 MF–68 Minifloppy Disk System or DMAF–1 Large Floppy Disk System; 1 RICHO Printer (daisy wheel) or QUME Printer (daisy wheel) or CENTRONICS Printer (dot matrix) or DIABLO Printer (daisy wheel) or any other Printer with full column width and with proper lower case letters.

The extra memory is useful for handling quantities of text in the region of more than five pages of A4 at a time, but is not by any means essential.

The difference between VDU's lies in what they will do, how much they will display, the quality of the keyboard, and the quality of the casing. The last two have a direct relation to price, but the first two do not except within one retailer's product range.

The mini floppy disks hold 80,000 characters on each side and the large ones currently hold 300,000 characters.

The choice between printers lies between quality of print and speed of printing. Daisy wheel printers operate at between 40 and 55 characters per second, and provide a choice of good quality print fonts. Some of them offer the ability to produce *proportionally* spaced output, though, at the time of writing, the Word Processor does not support this feature and would have to be extensively rewritten. Proportional spacing means that letters such as 'i' take up less room than letters such as 'w'. Dot matrix printers operate at speeds from 30 to 240 characters per second, and though the quality of print can be quite high, it does not compare with that from a daisy wheel printer. A further option is to use a modified IBM golfball typewriter. This has the advantages of cheapness, and a choice of IBM type fonts, but the speed is only 30 characters per second. An IBM service contract can be arranged to cover the typewriter, but with its dependence on many mechanical parts, it has not proved to be robust enough to stand up to heavy computer use. To be fair, we did not have a maintenance contract.

Entering the text

Text is entered under the operation of 'Text Editor'. This is a comprehensive editor that has a great many options and commands, but it is quite possible to use it and use only a limited range of these commands. I am not going to describe the operation of the whole computer system as a computer, so we shall have to accept that the machine is running and ready for use by a secretary. The first thing you do is summon Text Editor. from the disk by typing: EDIT LEGAL1, where LEGAL1 is the name of the document being created. I am going to take the example of a Conveyance of property as this represents a rather complex document, and gives me the opportunity to show some of the special features of this system. You should not imagine therefore that the system is designed for solicitors; it could be of use in *any* office. The secretary then begins to type, and let us suppose that you type the following:

```
 1.00=LEGAL1
 2.00=
 3.00=THIS CONVEYANCE is made the
 4.00=Fifth day of November
 5.00=One thousand nine hundred and seventy eight
 6.00=BETWEEN
 7.00=CHARLES SWEETEN
 8.00=of 18 South Road Oundle Peterborough
 9.00=(hereinafter called "the Vendor")
10.00=of the one part and
11.00=ARTHUR BELVEDERE CRUNCH
12.00=of 27 Shingles Way Cambridge
13.00=(hereinafter called "the Purchaser")
14.00=of the other part
15.00=WHEREAS the Vendor is seized of the property
16.00=hereinafter described for an estate in
17.00=fee simple free from incumbrances subject
18.00=only as hereinafter mentioned and has agreed
19.00=with the Purchaser for the sale to him
20.00=of the said property at the price of
21.00=twenty five thousand pounds
22.00=NOW THIS DEED WITNESSETH as follows:—
```

. . . . and so on, until you are done, when you will type # at the beginning of the line. This brings you out of the entry mode and into the editing mode.

While you are typing, you will make mistakes, and you will correct these by backspacing and retyping. But there will be later corrections and amendments. For example you may wish to alter something in line 15. To do this there is a Change command

15C/seiz/sies/

where '/' is used as a separator, and '15' indicates that the command should operate on line 15.

In order to inspect the text it is only necessary to type a command such as: 10P15, which will cause 15 lines to be typed, starting at line 10.

You will notice that the lines are of uneven length, and a solicitor would also tell you that the final document must be typed without leaving spaces. The ability to cope with this automatically is one of the attractions of this system which will be described later under 'Word Processor'.

In documents such as these there are a number of paragraphs (clauses) that are to some extent standard. These can be called from the disk file system and, if any modification is needed, they can be modified. For example:

READ	you ask to load
TAPE OR DISK (T-D) ? D	you reply D for disk
FILE NAME? CL184	you supply the name of the clause you want

The clause will now be at the bottom of the file. A typical clause that might be wanted quite frequently is:

> The Purchasers so as to bind the property hereby conveyed and every part thereof jointly and severally COVENANT with the Vendor that the Purchasers and their successors in title will at all times hereafter observe and perform the covenants contained in the Second Schedule to the said Conveyance dated the
>
> One thousand nine hundred and seventy eight

It is easy to insert the correct date into the appropriate blank line.

What I am suggesting is that complicated documents such as a Conveyance, can well be put together from a *bank of clauses* held on disk files, leaving only the one-off clauses to be typed.

Suppose now that you decide that the 12 lines which make up clause 3, starting at line 103, should be deleted. This is easily done by: 103D12

And suppose that the 22 lines starting at line 214 should be moved down by 31 lines. This is easily done by: 214MO 31 22

You now wish to 'Save' this typing on the disk, and this is done by typing S. This will result in a file being created on the floppy disk which will be called LEGAL1. The disk can be labelled and filed, and as part of the operating system of the computer, a catalogue of the contents of the disk can be printed out automatically to go in the manual filing system.

You may then Edit another file or the same one again. If you Edit the same one, the machine will automatically set aside a copy of the file as it then exists. Each time you Edit the same file, the machine will delete the previous backup file and replace it with the current file.

Suppose at some later stage the whole transaction gets postponed from the 5th November to the 16th of December; or it might be that you wanted a similar document for another house on the same housing estate. The date might perhaps occur at several points, and you wish to change them all. Obviously you could go through changing each occurrence as you found it under the operation of the Text Editor, but as a small example of how the full range of commands make the task of editing simpler, this is all you need to type to change all occurrences:—

C/Fifth day of November/Ninth day of December/!★

Yes, I agree it looks complicated, but it is doing something quite complicated, and you don't *need* to use it. Many businessmen have been surprised to find that their secretaries are often exceedingly intelligent, and if they care to demonstrate the simple use of the machine, and then leave the manual in a handy place, they might well find their secretaries becoming more proficient than themselves!

In order to obtain a printed copy of the file you will merely type: P LIST LEGAL1 and switch on the printer. This is in itself a form of word processing as described so far, in that it enables typed material to be stored, edited, amended, and typed as often as desired.

This paragraph should be skipped by those who are unfamiliar with Editors. TSC Text Editor (which is what I have been describing) is a pretty good editor, but it does have its failings in that it lacks some features to be found on others. For example: there is only one edit buffer; no macro can be defined (this is the most serious defect; there are several facilities that are desirable, for example looping, and conditional termination of loop); there are no character orientated commands — like 'move 3 positions' — 'insert CRLF'. The lack of the latter is quite infuriating at times. However it does have features which are not always found elsewhere. For example: the < line > directive < target > structure which allows a command to take effect over a specified range; line overlays; settable tabbing; definable special characters; and of course, the one that I regard as indispensible, the *move* command. Recently, a much more powerful Editor has been developed by SOURCE, which is a new software house started by ex-pupils of Oundle School. This Editor has most of the desirable features referred to above.

It would not be worth considering such a system as I have outlined for what has been described so far, though there is no doubt that your secretary would thank you.

One feature of this system which is good is that having started the machine on printing out a finished file, you may then start to edit the next file while the printing is still going on. But the major attraction comes when you examine the options offered by the part of the system known as 'Text Processor'.

Printing the text

The 'Text Processor' is a very complex program which gives the user a staggering degree of control over the exact layout of the printed output from this system. It works on the principle that there is a default mode of operation unless the user specifies otherwise. This means that you do not *have* to know anything about it at all. However, in this case, a little knowledge is a useful thing.

What can be done with Text Processor? You may specify the exact fitting of the page; this means that you can decide on the margins, and the length of text on each line and on each page. You can do automatic numbering of the pages and put in page titles and text headings. You can perform left and right justification of the text, so that both the left and right margins are straight. You can centre text lines automatically. You can define combinations of commands and refer to these by your own command. This last feature is particularly useful as it enables a relatively inexperienced user to use complicated procedures that someone else has worked out, and remain unaware (and not frightened) of the complexities involved. There are several more features that are available, but they are successively more difficult to describe without lengthy demonstration.

One more feature is worth mentioning though, and that is the ability to give a reference to a footnote, follow that with the footnote itself (which is the logical way to write it), and end up with the footnote fitted in correctly at the bottom of the correct page.

If you read that last paragraph again slowly and then think about what you can actually do with these features, you will realise that you are well on the way to being able to produce text to the standard produced by a professional printer. The limitations at present consist of proportional spacing and the range of character fonts and point size of lettering. The first of these will inevitably be done by someone soon. The second depends on the printer manufacturers, and the third has reached the stage where certain dot matrix printers will print double height lettering.

I shall now give examples of what I have outlined above. It is likely that I shall not see proofs of what I have provided for the printers of this article and so everything will depend on their type-setting. Naturally, I am using the 'Text Editor' and 'Word Processor' to produce it exactly as I want it as copy for the Editor of **PCW**.

In order to get the 'Word Processor' to take special action, for example on titles and page length etc., it is necessary to give commands that refer to particular sections of the text. This is done by inserting the commands *into the text itself.* They are distinguished from the text by occupying a line of their own, and by marking that line by a full stop in the first character position (it is unlikely that you would want to have a full stop here under any normal circumstances).

So for example, you might decide that the line length of the document given above was to be 60 characters, and that the page length was to be 50 lines. So you would insert two lines at the beginning of your text:

```
.LN 60      line length = 60
.PL 50      page length = 50
```

Please note that the comments on the right are NOT required, but are put in for ease of understanding in this description.

Legal documents that I have seen do have a habit of starting at the top, and continuing to the end without punctuation, paragraphs or pause. This means that no more commands are necessary for the 'Word Processor' which will now be able to output the document with everything in its place, and with both margins straight, and with no gaps in the text. This would output the text shown earlier in the following kind of format. The length of line would default to 60 characters, but I have shortened it to 40 to fit the PCW column.

> THIS CONVEYANCE is made the Fifth day of November One thousand nine hundred and seventy eight BETWEEN CHARLES SWEETEN of 18 South Road Oundle Peterborough (hereinafter called "the Vendor") of the one part and ARTHUR BELVEDERE CRUNCH of 27 Shingles Way Cambridge (hereinafter called "the Purchaser") of the other part WHEREAS the Vendor is seized of the property hereinafter described for an estate in fee simple free from incumbrances subject only as hereinafter mentioned and has agreed with the Purchaser for the sale to him of the said property at the price of twenty five thousand pounds NOW THIS DEED WITNESSETH as follows:—

Those of you who are more ambitious in the way that you write may wish to read on. You will want to centre your title, and under-line it, in the middle of the line. So you insert a command in front of the title, and follow the title with 3 blank lines by inserting another command:

```
.CE 2       centre the next two lines
CONVEYANCE for SALE OF LAND
- - - - - - - - - - - - - - - - -
.SP 3       space down three lines
```

Now you want to start your paragraphs by leaving two lines blank and by indenting the first line 6 spaces. To avoid putting the commands for this in each time, you define your own composite command (known as a macro) to do what you want. In this case I shall call the macro by the name '.PP' and I shall use the command for a single line indent.

```
.DM PP      define the name 'PP'
.SP 2       leave 2 lines blank
.SI 6       indent 6 spaces
..          end of macro command sequence
```

So in front of your paragraph start you insert your new command like this:

```
.PP
1) IN pursuance of the said agreement and in
consideration of the sum of
TWENTY FIVE THOUSAND POUNDS
paid by the Purchaser to the Vendor
(the receipt of which sum the Vendor hereby
acknowledges) the Vendor as beneficial owner
HEREBY COVENANTS
etc
```

Assuming a line length of 40 characters (it has to fit inside a PCW column width!), the output could look like

CONVEYANCE for SALE OF LAND

> THIS CONVEYANCE is made the Fifth day of November One thousand nine hundred and seventy eight
> BETWEEN
> CHARLES SWEETEN of 18 South Road Oundle Peterborough (hereinafter called "the Vendor") of the one part and
> ARTHUR BELVEDERE CRUNCH of 27 Shingles Way Cambridge (hereinafter called "the Purchaser") of the other part

WHEREAS the Vendor is seized of the property hereinafter described for an estate in fee simple free from incumbrances subject only as hereinafter mentioned and has agreed with the Purchaser for the sale to him of the said property at the price of
twenty five thousand pounds

NOW THIS DEED WITNESSETH as follows:—

1) IN pursuance of the said agreement and in consideration of the sum of
TWENTY FIVE THOUSAND POUNDS
paid by the Purchaser to the Vendor (the receipt of which sum the Vendor hereby acknowledges) the Vendor as beneficial owner HEREBY COVENANTS
etc

The manual that describes the 58 different commands and the 26 registers and the 7 special characters has been written with a degree of conciseness that does not make things easy. Each command gets an average of four lines of description, and you have to look elsewhere for the all too rare examples. However they have provided a 'standard set' of commands to deal with footnotes, two column output, and form letters.

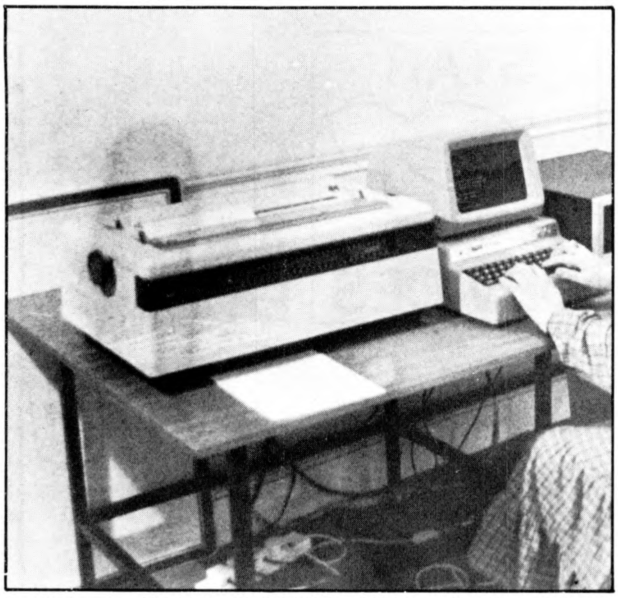
If you intend to do something else which is not very simple indeed, then you would be wise to obtain help, or expect to take some time in mastering the difficulties. It is worth quoting the authors, Technical Systems Consultants, on the subject. "The TSC Text Processor is the most complex program released by TSC to date. Do not expect to master the system with one reading of the manual. The entire document should be read lightly the first time through, followed by more rigorous reading. Many results may occur which are contrary to the user's intentions. If strange output is encountered, reread the manual." And most software distributors go out of their way to tell you how easy it is to use! Fortunately the difficulties only start when you try to be clever, and so far I have yet to meet anyone who has not finally realised that the Processor does exactly what it is told to do.

Conclusion

The cost of the programs 'Text Editor' and 'Word Processor' is 25 pounds each — not expensive. The cost of a mini-disk computer system is about 1900 pounds plus printer. The computer and the programs are available from Computer Workshop who in turn have a number of agents in the UK. The printers can be bought from any source which offers a good price, though there is something to be said for the original manufacturer.

The system works well, and offers considerable scope for the computer to help in other areas of a business. But I regard the keyboard/terminal as supplied from SWTP as sub-standard and only to be tolerated if you cannot afford a better one. Replacing it with say, a LYME would raise the price by 200 pounds. I have only seen the system with a RICOH daisy wheel printer attached, and so unless the retailer can demonstrate another daisy wheel actually in use on the system, I would recommend that. Dot matrix printers are fairly simple to connect, but again I would recommend seeing them attached before placing an order.

This must represent outstanding value for money as a Word Processing system. I understand though that there will be a similar type of system which operates on those machines which have the SC/M operating system. As is usual then, I must advise anyone contemplating a purchase that they would be well advised to wait for a year. At which time I will again advise them to wait for a year. And so on.



Example of SMALL BUSINESS APPLICATION: SWTPC Computer used as a word processor.

BUSINESS COMPUTING

Part 1

Rodnay Zaks

INTRODUCTION

For the first time the progress of technology makes it possible to enjoy the benefits of a computer in a small business environment for less than five thousand pounds. *Or is that really true?* The answer is "yes, but ..." The purpose of this article is to justify the "yes" and to describe the "but".

Can a low-cost microcomputer system provide true business computing capabilities? Yes.

Is there any one system presently available that does? No.

It will be seen that the essential deficiency of actual microcomputer systems is not at the *hardware* level but at the *software* level. This has always been the case, ever since computers were introduced, and history has consistently repeated itself every time a new generation of hardware was introduced. It will be seen that the necessary hardware to process efficiently a number of business applications can indeed be purchased for £5,000 to £20,000. However, software is just beginning to become available. Naturally many trade-offs exist in function of the capabilities one wishes to acquire, and these will be studied.

Therefore, the classical applications of computers in business will first be reviewed, in order to define the processing capabilities required to achieve specific business goals. In order for the businessman to make a reasonable choice of a computer system, *it is imperative that he understands* the trade-offs between the various solutions available today as *there is no "best".* The choice can be somewhat compared to the selection of a new car or of a new complex machine in function of a specific intended application. There is no general-purpose choice fit for all applications.

Understanding the *hardware* required and the hardware available is a relatively simple matter. The more complex and difficult problem is understanding the *software* capabilities required. This is where a large majority of persons purchasing a business system make mistakes. These mistakes are generally more costly than hardware ones. Typically *software* investment in a system will quickly become the dominant one.

An inadequate system will limit the possible growth of the capabilities of the system, and possibly of the business. A transition to a different system might be costly and disruptive. For these reasons, the reader is strongly encouraged to study and understand the software concepts as well as the hardware ones that will be presented.

Applications of Computers in Business

Every business needs primarily to maintain a number of files. The best known files are: accounts receivable, accounts payable, inventory, general ledger. Additional files which are usually desirable are: personnel, customers list, mailing list, back-orders lists, sales list, vendors list, cash situation, company property, and more.

These lists are managed either by hand (typically by a bookkeeper), or with the help of electro-mechanical devices, or by computer, or by a combination of the above.

In addition to maintaining files, every business applies specific *processing* techniques to each of them. For example, a *payroll program* will process the *personnel file* and generate payroll reports, as well as print cheques. A *tax program* will process the *sales reports* and the *personnel files* to produce the required *tax reports.* A *transaction procedure* program will manage updates of specific files, and changes, or entry, of new data. A typical example is a new sale: the transaction program will utilize the inventory file, supplier file, customer file, and perhaps others. It will update them, and print reports.

Similarly an *incoming shipment procedure* will handle shipments coming in and will enter them in the inventory file, check for back orders, and add entries to the accounts payable list.

Any *payment received* will update the *accounts receivable* list and the *cash situation* list.

In addition to the main programs a number of additional programs must be available in order to produce useful reports. These additional facilities required will be described in more detail in the text following.

It is important to note that the *principle is quite simple:*

1- Files must be created and maintained.

2- Programs should be available to provide the interface between the user and the files, and supply the required processing functions.

Unfortunately in a real business system, this is only *part of the processing* required. In fact, in most businesses, the direct maintenance of a *single* file is reasonably simple. The bulk of the processing required is due to the simultaneous *cross-referencing and automatic updating* of multiple files.

Let us look at an example. An order is received by mail. It will be processed by the *transaction manager* program. The sale will be entered in the *sales file* for the day. A complex sequence of events now unfolds. As a result of this entry, the name of the customer will be added to the *custo-*

mers list automatically. In addition his name will probably be coded in function of the purchase he has made or of the amount of the purchase, or of his job position. In addition, his name will be checked for credit information before the order is processed. Provided the sale is not "vetoed" by the *credit manager* program, the next step is to honour the order. The *saleable inventory* file will now be checked for the availability of the items ordered. In this example three items are ordered: A, B and C. A and B are in stock. C is not.

As a result, an *invoice* to the customer is generated, a *shipping list* and a *back order* are generated. The back order is added to the *back order list.* In our example, item B is available in stock. The *inventory list* is structured with a special field which specified the re-order level. The re-order level of item B is four. As a further result of this transaction, a back-order or re-order will also be generated for item B for a standard quantity of 25 items (the number 25 was specified in the inventory file). The address of the vendor is obtained from the *vendors file* by using the vendor number as an index to the list.

This simple sales transaction has required the use of five files within our system and of several processing programs. For specific businesses, it might even be necessary to update, check, or modify additional files, or perform additional processing functions. It should be clear from this example that, in order to be truly useful, a business system must provide ways to **access, modify** and **process** conveniently a variety of files. In addition it must provide a mechanism for performing all the required functions automatically, not manually.

Unfortunately, it will be seen that *the majority of so-called business systems available today, using microcomputers, do not perform such a complete service.* They provide usually *single file* management and do not automate completely the complete transaction process. Much has to be done "by hand".

Word Processing

"Word processing" refers to computerized typewriter operation, where the user can easily change, modify, or format text. It requires an "editor" program, a standard facility of traditional computers. The cost of the processor has become so small that it can be dedicated to a function such as word processing so that "standalone" word processors are multiplying. The majority use a Selectric or similar typewriter. By contrast, business systems offer the option of displays or multi-terminals.

Using a Computerized Business System

Let us use now an in-house microcomputer for a simple transaction. We will specify the type of program, and our choices in response to choices or questions appearing on the screen of the CRT terminal.

Initially, the system displays a "menu". A "menu" is simply a multiple-choice question. The question asked by the system is stressed by one or more "prompt characters" (here, ".."), designed to indicate that the microcomputer is waiting for an answer.

Fig. 1 A "MENU"

The "business program" has been selected. The system should load it automatically from the disk. A directory of options appears again.

Fig. 2 THE BUSINESS "SUBMENU"

We specify the "accounts receivable". At this point, the system may request that a new diskette be inserted. Let us assume not, and proceed.

Fig. 3
THE ACCOUNTS RECEIVABLE FILE

We specify a new sale, and the system will request all data needed to record the transaction, generate an invoice, and later update all related files such as bank, accounts receivable, inventory, customer list. The dialogue becomes now highly interactive with the system requesting all necessary data.

Fig. 4 ENTERING A NEW SALE

Fig. 5 SALE ENTRY, CONTINUED

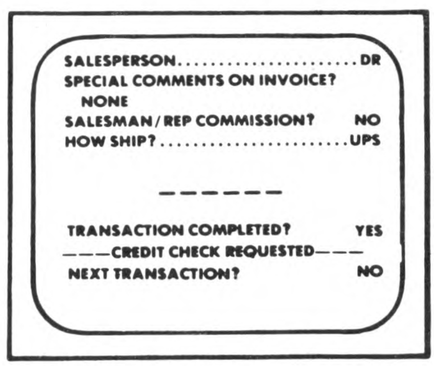


Fig. 6 SALE ENTRY, END

The transaction is now completed.

The mode of interaction with the system should now be clear. The program asks all necessary questions, enforcing a discipline. In addition, we will see that it should also check the validity of data being entered (no gross errors). Finally it should automatically print invoices, and later update all related files.

Let us now examine in more detail the actual requirements.

The Requirements of a Business System

The requirements of a business sys-

tem will be analysed here in terms of the essential *files* that must be maintained and of the essential *processing functions* which must be performed.

Accounts Receivable

This is essentially the file which contains a copy of all invoices generated by the system. Naturally the file does not contain the actual copy, but the minimum amount of information that it is possible to store, which allows the system to actually generate a complete invoice. Typically, it will store the date of the transaction, the name and address of the customer, shipment point, sales information such as salesman, how shipped, when shipped, and specific details of the items sold. It may not be necessary to store all the information which appears in a usual invoice within this *accounts receivable* file. If a *sales file* exists, all this information is stored there, and one needs to be accessed *frequently* and which needs to be *processed efficiently.*

Efficient information processing by any computer requires that all elements within a file be of *equal length.* For this reason, all files which are processed often, or by complex programs, use *fixed length* entries or "blocks". An accounts receivable file can be structured in that manner. Fixed fields can be allocated to essential information such as date, name, amount due, transaction or customer code, invoice number. The presence of the invoice number allows the user of the system to access the remainder of this information in the *sales list* or in the *invoice file.* In computer jargon, the presence of a number used to access information stored elsewhere is called a *pointer.* The invoice number is a pointer to the actual invoice. In business jargon, this is part of the *audit trail.*

The accounts receivable file must be distinguished from the accounts receivable program. The accounts receivable file is simply the list of accounts. The advantages or disadvantages of its format are easy to evaluate by the business user. A typical requirement is that it contain, in an easily accessible way, all the fields that the business user requires frequently.

The accounts receivable *program* is responsible for manipulating this file, updating it, and generating the required report. It must also generate specialised reports such as the printing of accounts older than 30, 45, 60 or 90 days (this is called "aging"). This program can be even responsible for generating automatically *reminder notices.* However, the reminder notification program may be a separate program. In this case the accounts receivable program would be used for generating a file of *overdue accounts.* This file would then be used in turn by the *reminder notice program* in order to generate personalised reminders to all customers listed in the overdue file. Whether to separate functions into individual programs or integrate them within a single program has little impact on the value of this system. It is largely a matter of programming convenience for the system designer. The important point is that *all the facilities be available.*

Accounts Payable

The accounts payable file is essentially a list of all bills or invoices received by the business. Typically, whenever an "OK to pay" order has been entered, the accounts payable manager program will automatically print payment cheques for the goods received. Typically, the cheque will be printed either at a specified date, or else at a programmed date such as thirty days after receipt of invoice. (A good cheque printing program should also check that the cash balance in the bank account is sufficient to cover the expenditures!)

Inventory

There is no optimal inventory file, as inventory information is different depending on specific business needs. For this reason, most general purpose inventories files will carry a large number of categories. Not all categories will be used by the business. The unavailability of some categories can be felt to be a drawback by some users. The availability of too many categories on the other hand, means that a significant amount of space is wasted in the system. This translates into a relatively smaller number of items that may be entered in the inventory. However, with the ever decreasing costs of memory, the clear trade-off now is to provide as many categories as possible, for most types of businesses, even if some of them are never going to be used. It should be remembered that the size of the inventory file is limited by the physical storage available, such as the size of a diskette.

Typical information which may be included in an inventory file is the following:

CODE – ITEM NO. – ITEM DESCRIPTION – STORAGE LOCATION – NUMBER AVAILABLE – VENDOR NUMBER – FILLING PRICE – PURCHASE PRICE – LAST SALE DATE – MINIMUM QUANTITY FOR RE-ORDER.

Typically 64 to 128 bytes at a minimum must be provided for such an entry. Using such a format, 1800 to 3600 items may be stored in a typical diskette.

The inventory control *program* must provide many functions. It must provide generalised inventory management facilities:

- complete inventory maintenance, including automatic updates of any category of information within the file.
- sales order entry
- purchase order entry
- sales history
- automated backorders
- list of quantity, class, cost, vendor, item no., date of sale
- minimum quantity search
- selective update
- activity reports
- inventory lists in functions of combinations of criteria.

As a rough indication, a minimal inventory management, written in BASIC will require 10K words of memory (for all practical purposes a "word" is a "byte" here, in the case of 8-bit microprocessors). A more general program will easily require 90K or more. Since the central memory of a microprocessor is never larger than 64K, an *overlay* technique is used, so that such large BASIC programs can be run on a smaller main memory. An *overlay* consists in executing one part of the program, then bringing in the memory an additional part of the program and overwriting a no-longer-required segment of the previous one which had been installed in the main memory, and so on. The complete BASIC program is therefore never completely resident in the memory in one piece. Pieces of it are brought into the central memory as needed. Naturally this reduces the efficiency of the processing. However, if the overlays are cleverly written, the impact on efficiency is reasonably small.

Update

It is important to note once more, that, technically, update on an inventory file can all be performed by hand. The user can examine the list of items in the inventory and modify any of the entries such as the unit cost. However the real value of the computer system is again in automating the updating of identical information in many files. Therefore a comprehensive business system should automatically update the inventory file, whenever relevant information is changed somewhere else. For example, should the unit cost of the product be changed, it should be updated automatically in the inventory file as well as in any other file where it might reside.

PCW *To be concluded in the next issue. Extracted from the author's book, "An Introduction to Personal and Business Computing", published by* **Sybex, 313 rue Lecorbe, Paris, France.** *Highly recommended.* **PCW.**

PET Preening

Gordon Bell, *Micro Systems*

One interesting feature of the PET is the real time clock which runs continually all the time the PET is switched on. The time is easily obtained via the Basic programming language by accessing the variable TI$.

The following program allows the PET to behave as a timing device with specific tasks being undertaken automatically at specified times of the day. The times at which events are to occur are read into the program from a DATA statement and stored in array A$ (a rogue value of 9999 is used to terminate the data). The real time clock is then accessed and the time obtained is compared to the times stored in the array A$. If a match is found then a given subroutine will be performed and control returned so the time comparisons can continue. The program shown is given as a basic skeleton which readers can modify for their own purposes by writing their own subroutines and entering them at the appropriate locations in the program.

I believe most readers will have their own ideas regarding what they would like to do within the subroutines rather than the more obvious ones of printing messages on the screen or turning external devices on or off via the user port.

Finally, before running the program, the programmer must remember of course to initialise the real time clock with the correct time as described in the PET handbook.

[Readers of my article in the October issue please note that there should be two brackets, viz.)) after RND (3].

```
10    DIM A$(10)
20    DATA 1600, 1601, 1602, 1603,
      1604, 1605, 9999
30    X = X + 1
40    READ A$(X)
50    IF A$(X) <> "9999" THEN 30
60    B $= LEFT $(TI $, 4)
70    FOR Y = 1 TO X
80    IF B $<> A $(Y) THEN 100
90    Z = Y
95    A $(Y) = "0000"
100   NEXT Y
105   IF Z = 0 THEN 60
110   ON Z GOSUB 200, 400, 600,
      1200
115   Z = 0
120   GO TO 60
200   GOSUB 1400
210   RETURN
400   GOSUB 1400
405   RETURN
600   GOSUB 1400
610   RETURN
800   GOSUB 1400
810   RETURN
1200  GOSUB 1400
1210  RETURN
1400  PRINT "TIME = "; B$
1410  RETURN
```

PCW The author may be reached at 55 Belvedere Road, Hessle, North Humberside **PCW.**

DRAWPIC

Taking the sweat out of Computer Graphics

A. O. Ellefsen

The following very short and simple program (see listing) was written on the Tandy TRS 80 Level Two 4K Micro Computer whilst working on a major printed circuit design project.

It will give the kids something to play with (not to mention the mums and dads) and does form the basis for more serious work.

The object of the exercise is to provide a means of drawing pictures on the VDU using the "INKEY$" statement which enables your program to be manipulated *whilst executing.* The system commands are:

T = Trace (draw a line on the screen)
C = Cursor (move a cursor about on the screen)
L = Move one increment left
R = Move one increment right
U = Move one increment up
D = Move one increment down
M = Reproduce pattern from memory
W = Wash out current pattern
E = Erase screen
Z = Zero memory

The size and complexity of patterns that can be stored are determined by the amount of memory available and is fixed in line 4. A running check of how much room is available is printed out continuously in the bottom right hand corner of the screen prefixed by 'N', also the current cursor/trace position prefixed respectively by 'X' and 'Y'. To give an absolute check on memory add on to lines 58 and 1Ø55 "MEM"; MEM; and make use of the TRS80's 'Print Mem.' command.

The top left hand corner of the screen reminds you which of the two major command modes you are in by printing "Trace" or "Cursor". Make sure in line 19 that a space occurs between the 'E' of TRACE and the final inverted commas. Otherwise, after passing through the 'CURSOR' mode, the final 'R' of CURSOR will appear at the end of TRACE resulting in 'TRACER' thereafter.

I suggest that lines 2Ø and 1Ø2Ø be written *exactly* as shown without modification. Some very odd effects can occur if this is not done. The rather cumbersome method of writing the Trace module lines 19 to 6Ø and the Cursor module lines 1Ø19 to 1Ø6Ø is directly attributable to this problem.

Lines 3ØØØ and 3ØØ1 ensure that when reproducing patterns using the 'M' command the starting point of the pattern is linked directly to the current cursor or trace position. Take care when moving the cursor about that you do not let it erase parts of your existing display. A bit of effort and ingenuity could probably avoid this problem by using the Tandy 'POINT (X,Y)' command which detects if a particular section of the screen is already 'occupied'.

On first running the program one enters the Trace module. If cursor control is required hit 'C' but on first time round also hit any key other than 'Break', or any of the command keys to ensure that when you move the cursor away from the centre of the screen, a spot does not remain illuminated in the centre of the VDU field.

After producing a pattern on the screen it is important to remember that when reproducing this pattern its starting point will be from the current cursor position. Therefore, it is advisable to go into the cursor mode before hitting "M". Check that the cursor position is known by moving it in the appropriate direction to avoid erasing any of the existing pattern. If, accidentally, any of the existing pattern is erased, restore the cursor to its original position and hit "M".

Hitting break and entering, in the instantaneous mode:

```
FOR N = 1TO (C–1): PRINT Z(N);W(N);: NEXT
```

enables one to inspect the X,Y coordinates of the current pattern. After one or two experiments performed in this manner one realises that it gets rather boring when one is constantly retyping the same line again and again. A fairly obvious solution is to write new lines between say 1Ø45 and 1Ø55 such as, for example:

```
1Ø46 IF A$ = "5" THEN 5ØØØ ..... then adding the punch line ..... 5ØØØ CLS:FOR N=1 TO (C61):PRINT Z(N);W(N),: NEXT
```

I have a feeling that this would be useful if you need to do any debugging or, more importantly, if you are using a machine other than the TRS80 which may have a different Basic dialect.

It will be quickly seen that for any given pattern the coordinate listing remains constant wherever the pattern is generated on the screen. Substituting F(N) for W(N) and G(N) for Z(N) in 5ØØØ lists the coordinates for specific placements on the screen.

Once you start experimenting you can reach for the sky. For example, try the following. But make sure, if you're using a 4K system, that you have sufficient memory available. A certain amount of trimming in line 4 helps.

```
6ØØØ FOR N=1 TO (C–1) : SET (Z(N) /2,W(N) /2): NEXT: GOTO 1Ø19
```

When my daughter saw 6ØØØ being executed she exclaimed "Ooo ....! look, it's having a baby". Hence routine 8ØØØ – make sure your patterns don't exceed the I increment of 1Ø.

```
8ØØØ FOR N= 1 TO (C–1) : IF (F(N)+I) 127 THEN 8Ø2Ø
8ØØ5 SET (F(N) +I,G(N) )
8ØØ6 NEXT
8Ø1Ø I=I +1Ø : GOTO 8ØØØ
8Ø2Ø I=Ø : GOTO 1Ø19
```

When running this always hit "M" when in the cursor mode before "8". It takes little imagination to extend this proliferation into the Y axis. Who said computers were sexless!

By this time your finger will almost be dropping off with pushing the cursor around the screen, so try this one:

```
9ØØØ PRINT @ Ø, "ENTER X,Y COORDINATES";
9Ø1Ø INPUT X,Y
9Ø2Ø GOTO 19
```

One could go on and on. Don't forget your entries:

```
1Ø47 IF A$= "6" THEN 6ØØØ
1Ø49 IF A$= "8" THEN 8ØØØ
1Ø5Ø IF A$= "9" THEN 9ØØØ
```

I hope you will excuse my liberal extension into high line numbers. This is engendered by the fact that the level 2 TRS80 allows up to 65,529 of them; but, as a final thought, keep a few available for subroutines such as a large alphabet. Use the INKEY$ command and all the letters A to Z but enter them into the program whilst holding down the shift key so as not to confuse them with the existing commands. You will require some form of incrementing as listed under 8ØØØ and further incrementing in the Y direction at the end of each line at the same time setting I to zero. For each letter generated you will need to substitute variable names for W(N) and Z(N) but this is no problem as Level two has plenty of these – in the neighbourhood of 900.

```
2     CLS
3     CLEAR
4     DIM F(13Ø),G(13Ø),Z(13Ø),W(13Ø)
6     N = Ø
1Ø    X = 64:Y=23
19    PRINT @ Ø, "TRACE ";
2Ø    A$= INKEY $: IF A$=""THEN 2Ø
25    IF A$= "W" THEN 4ØØØ
3Ø    IF A$= "L" THEN X=X–1
4Ø    IF A$= "R" THEN X=X+1
42    IF A$= "U" THEN Y=Y–1
43    IF A$= "D" THEN Y=Y+1
47    IF A$= "E" THEN CLS:N=Ø:GOTO 2Ø
48    IF A$= "Z" THEN 2
49    IF A$= "M" THEN 3ØØØ
5Ø    SET (X,Y)
51    IF A$= "C" THEN 1Ø19
52    N=N+1
53    W(N) =Y:Z(N) =X
55    C=N+1
58    PRINT @ 1ØØØ,"X";X;"Y";Y;"N";N;
6Ø    GOTO 2Ø
1Ø19  PRINT @ Ø,"CURSOR";
1Ø2Ø  A$= INKEY $: IF A$= ""THEN 1Ø2Ø
1Ø25  Q=Q+1:IFQ<=1 THEN 1Ø25
1Ø3Ø  IF A$="L" THENX=X–1:SET(X;Y):RESET(X+1,Y);
      IF Q=2 THEN SET (X+1,Y)
1Ø4Ø  IF A$= "R" THEN X=X+1:SET (X,Y):RESET (X–1,Y):
      IF Q=2 THEN SET (X–1,Y)
1Ø42  IF A$= "U" THEN Y=Y–1:SET (X,Y):RESET (X,Y+1):
      IF Q=2 THEN SET (X,Y+1)
1Ø43  IF A$= "D" THEN Y=Y+1:SET (X,Y):RESET (X,Y–1):
      IF Q=2 THEN SET (X,Y–1)
1Ø44  IF A$= "M" THEN 3ØØØ
1Ø45  IF A$= "T" THEN 19
1Ø55  PRINT @ 1ØØØ, "X";X;"Y";Y;"N";N;
1Ø6Ø  GOTO 1Ø2Ø
3ØØØ  E=X–Z(1)
3ØØ1  U=Y–W(1)
3ØØ5  FOR N=1 TO (C–1) :F(N)=Z(N)+E:G(N)=W(N)+U:NEXT
      N = Ø
3Ø1Ø  FOR N=1 TO (C–1) :SET (F(N),G(N) ) :NEXT
3Ø15  N =Ø
3Ø16  Q = Ø
3Ø2Ø  GOTO 1Ø2Ø
4ØØØ  FOR N=1 TO (C–1) :RESET (F(N),G(N) ) :NEXT
4Ø1Ø  GOTO 2Ø
```

Getting It Together

Build your own assembler — listing (concluded)

Mike Banahan

```
1175                         I        WORDTB IS USED TO LOOK UP A PARTICULAR
1175                         I        CHARACTER (SEE WOTSIT) AND FIND OUT BY LOOKING
1175                         I        AT THE BITS SET IN THE CORRESPONDING BYTE
1175                         I        WHETHER IT'S A TERMINATOR, ALPHABETIC AND SO ON
1136
1136 0000000000000000000 WORDTB: .BLKZ 11‡I         TO BEFORE TAB
1141 20010000                    .WORD 40,1,0,0
1145 01                          .WORD 1‡I           OR
1146 00000000000000000000        .BLKZ 22‡I          UP TO <US>
1158 0120010102020200101         .WORD 1,40,1,1,2,40,40,1,1,1,5,5‡I          UP TO +
1164 010502054848484845          .WORD 1,5,2,5,110,110,110,110,110,110
116E 484848484811                .WORD 110,110,110,110,21‡I        AT:
1173 0101110121                  .WORD 1,1,21,1,40‡I        AT ?
1178 2042424242424242            .WORD 40,102,102,102,102,102,102
117F 0202020202020202            .WORD 2,2,2,2,2,2,2,2‡I
1187 020202020202                .WORD 2,2,2,2,2,2
118D 020202020202                .WORD 2,2,2,2,2,2‡I        Z
1193 2020200101                  .WORD 40,40,40,1,1‡I        <-
1198 00000000000000000000        .BLKZ 40‡I          THE REST
11E8                     TERMIT: .BLKW
11E9                             SYMLIM=17777‡I  SYMBOL TABLE TOP LIMIT
11E9                     INSBLK: .BLKW 100
11F9                     INSCNT: .BLKW
11FA                     CHKSUM: .BLKW
11FB 0000                BOTADD: .ADDR 0
11FD                     OUTINS: ‡I        OUTPUT AN INSTRUCTION
11FD E5D5                        PUSH HL‡PUSH DE
11FF C5F5                        PUSH BC,AF
1201 4F                          C<-A
1202 2A0815                      HL<-CURADD
1205 3A0701FE0328029             A<-PASFLG‡CMP≠3‡BR Z,OUT.P3
120C FE012822                    CMP≠1‡BR Z,OUT.P1
1210 3AEC12FE0528 0F             A<-INSNUM‡CMP≠CWIDTH‡BR Z,OUTP20
1217 3E20CD070E                  A<-≠SPACE‡CALL CHOUT
121C F1                          POP AF
121D CDC81221EC1 34              CALL PR2‡HL<-≠INSNUM‡INC (HL)
1224 1801                        BR OUTRET
1226 F1                  OUTP20: POP AF
1227 2A0815232208 15     OUTRET: HL<-CURADD‡INC HL‡CURADD<-HL‡I  UPDATE CURRENT ADDRESS
122E C1D1E1C9                    POP BC,DE,HL‡RET
1232 F118F2              OUT.P1: POP AF‡BR OUTRET
1235                     OUT.P3: ‡I        OUTPUT INSTRUCTIONS TO TAPE
1235 3AF911FE0ACC6E12            A<-INSCNT‡CMP≠10‡CALL Z,OUT.PT‡I        WRITE A BLOCK?
123D 3AF911                      A<-INSCNT
1240 B72006                      TSTA‡BR NZ,.+11
1243 2A08152 2FB11               HL<-CURADD‡BOTADD<-HL
1249 16005F                      D<-≠0‡E<-A
124C 2AFB1119                    HL<-BOTADD‡ADD HL,DE
1250 B7                          TSTA‡   IHL CONT'S WHAT WE THINK IS
1251                                     ITHE CURRENT ADDRESS
1251 ED5B0815                    DE<-CURADD
1255 ED52                        SBC HL,DE
1257 2804B7                      BR Z,.+6‡TSTA‡  INEED WE BOTHER PRINT?
125A C46E12                      CALL NZ,OUT.PT
125D                                     ITHY'RE THE SAME
125D 3AF911                      A<-INSCNT
1260 16005F21F911                D<-≠0‡E<-A‡HL<-≠INSBLK
```

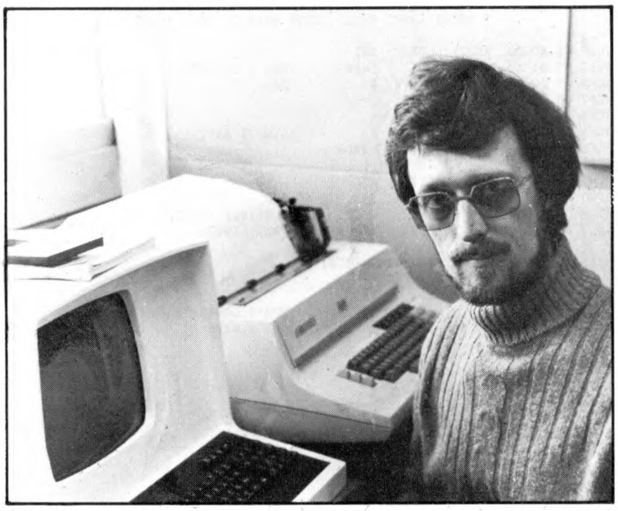
```
1266 19                          ADD HL,DE
1267 713C32F911                  (HL)<-C‡INC A‡INSCNT<-A
126C 18C4                        BR OUT.P1
126E
126E F5E5C5              OUT.PT: PUSH AF,HL,BC‡I PRINT A BLOCK OF INSTRUCTIONS
1271 AF32FA113C                  CLA‡CHKSUM<-A‡INC A
1276 CD8812                      CALL OCHOUT
1279 3DCD8812                    DEC A‡CALL OCHOUT
127D CD8812                      CALL OCHOUT
1280 3AF91147                    A<-INSCNT‡B<-A
1284 C606                        ADD ≠6
1286 CD8812                      CALL OCHOUT
1289 2AFB11                      HL<-BOTADD
128C 7CCD8812                    A<-H‡CALL OCHOUT
1290 7DCD8812                    A<-L‡CALL OCHOUT
1294 AFB8                        CLA‡CMP B
1296 280A                        BR Z,.+14‡       IZERO LENGTH (B=6)
1298                                              IMUST MEAN TRANSFER ADDRESS
1298 21F911                      HL<-≠INSBLK
129B 7ECD881223 10F9             A<-(HL)‡CALL OCHOUT‡INC HL‡DBNZ .
12A2 3AFA11ED44CD870E            A<-CHKSUM‡NEG‡CALL CHOUT
12AA 2A081522FB11                HL<-CURADD‡BOTADD<-HL
12B0 AF32F911                    CLA‡INSCNT<-A
12B4 C1E1F1C9                    POP BC,HL,AF‡RET
12B8 F5CD870E            OCHOUT: PUSH AF‡CALL CHOUT‡I    OUTPUT A BYTE, TALLY CHECKSUM
12BC 4F3AFA11                    C<-A‡A<-CHKSUM
12C0 8132FA11                    ADD C‡CHKSUM<-A
12C4 F1C9                        POP AF‡RET
12C6 7CCDCB12            PR16:   A<-H‡CALL PR2‡I PRINT 16 BITS IN HEX
12CA 7D                          A<-L‡
12CB F5                  PR2:    PUSH AF‡I        PRINT 8 BITS IN HEX
12CC 0F0F0F0F                    RRCA‡RRCA‡RRCA‡RRCA
12D0 CDD412                      CALL PR1
12D3 F1                          POP AF
12D4 E60F                PR1:    AND ≠17‡I        PRINT 4 BITS IN HEX
12D6 C630FE3A3802                ADD ≠60‡CMP ≠':‡BR C,PR1.1
12DC C607                        ADD ≠7
12DE CD870EC9            PR1.1:  CALL CHOUT‡RET
12E2
12E2                             ITO ADD ERRORS, DO AS BELOW R
12E2                             IINCREMENT ERRNUM DON'T FORGET ABOUT ERRPRN
12E2                     ATLONG: .BLKW‡I ATOM TOO LONG
12E3                     CHERR:  .BLKW‡I BAD CHARACTER READ
12E4                     SEQERR: .BLKW‡I PHASE ERROR
12E5                     SYNFLG: .BLKW‡  IERROR 7 SYNTAX ERROR
12E6                     TRERRF: .BLKW‡I TRUNCATION
12E7                     EVFATR: .BLKW‡I EVALUATION GONE WRONG
12E8                     MULERF: .BLKW‡I MULTIPLE DEFINITION
12E9                     EVERR:  .BLKW‡I EVALUATION ERROR
12EA                     LINLNG: .BLKW‡I LINE TOO LONG
12EB                     ONEERR: .BLKW‡  IERROR ≠1. INSERT OTHERS ABOVE TABLE
12EC                                     IEND OF ERRORS
12EC                     INSNUM: .BLKW
12ED                     SYNTER:
12ED F53E0132E512                PUSH AF‡A<-≠1‡SYNFLG<-A
12F3 F1C9                        POP AF‡RET
12F5                     TRERR:
```

```
12F5 F53E0132E612              PUSH AF*A<-#1*TRERRF<-A
12F8 F1C9                      POP AF*RET
12FD F5E5C5          ERRST:    PUSH AF,HL,BC*! CLEAR ERRORS
1300 AF21E212                  CLA*HL<-#<ONEERR-ERRNUM>+1
1304 060A                      B<-#ERRNUM
1306 772310FC                  (HL)<-A*INC HL*DBNZ .
130A C1E1F1C9                  POP BC,HL,AF*RET
130E
130E                 SYMNUM:   .BLKW
130F F5CDCC0E        SYMPRN:   PUSH AF*CALL CRLF*!     PRINT AND SORT SYMBOL TABLE
1313 CDA80E53594D424F4C        CALL PRINT*.ASCIZ%SYMBOL TABLE*,LF,LF,LF,CR
1327                                     !WARNING! WE LOSE REGISTERS
1327 DD21F415                  IX<-#SYMTAB*     !HERE COMES THE SORT
132B 110800                    DE<-#10
132E AF320F13                  CLA*SYMNUM<-A
1332 DDE5FDE1        SYMS.1:   PUSH IX*POP IY
1336 AFDDBE00282F              CLA*CMP 0(IX)*BR Z,SYMPRX
133C DDE5FDE5        SYMS.2:   PUSH IX,IY
1340 0606                      B<-#6
1342 DD7E00E67F      SYMLP:    A<-0(IX)*AND#177
1347 4FFD7E00E67F              C<-A*A<-0(IY)*AND#177
134D B93820                    CMP C*BR C,SWAP
1350 2006                      BR NZ,SYMS.3
1352 DD23FD2310F3              INC IX,IY*DBNZ SYMLP
1358 FDE1DDE1        SYMS.3:   POP IY,IX*      !GET START ADDR BACK
135C FD19                      ADD IY,DE*      !POINT TO NEXT
135E AFFDBE00                  CLA*CMP 0(IY)*  !DONE ?
1362 20D8                      BR NZ,SYMS.2
1364 CD9313                    CALL SYMPLP
1367 DD19                      ADD IX,DE*      !NOT FINISHED YET
1369 18C7                      BR SYMS.1
136B CDCC0EF1C9      SYMPRX:   CALL CRLF*POP AF*RET
1370 FDE1DDE1DDE5FDE5 SWAP:    POP IY,IX*PUSH IX,IY
1378 0608                      B<-#10
137A 21AC15                    HL<-#ATBLCK*    !ANYWHERE WILL DO ??!
137D DD7E00          SWAP.1:   A<-0(IX)
1380 77                        (HL)<-A
1381 FD7E00                    A<-0(IY)
1384 DD7700                    0(IX)<-A
1387 7E                        A<-(HL)
1388 FD7700                    0(IY)<-A
138B DD23FD23                  INC IX,IY
138F 10EC                      DBNZ SWAP.1
1391 18C5                      BR SYMS.3
1393 DDE5C5          SYMPLP:   PUSH IX,BC
1396 DDCB047E                  TEST 7,4(IX)
139A 3E4D                      A<-#'M
139C C4B70E                    CALL NZ,CHOUT
139F 3E20CCB70ECDB70E          A<-#SPACE*CALL Z,CHOUT*CALL CHOUT
13A7 0606                      B<-#6
13A9 DD7E00E67F      SYMP.R:   A<-0(IX)*AND #177
13AE 20023E20CDB70E            BR NZ,.+4*A<-#SPACE*CALL CHOUT
13B5 DD2310F6                  INC IX*DBNZ SYMP.R
13B9                           !IX NOW POINTING TO FIRST BYTE OF VALUE
13B9 3E20CDB70E                A<-#SPACE*CALL CHOUT
13BE DD7EFFE680                A<--1(IX)*AND#200
13C3 3E3A                      A<-#':

13C5 2802                      BR Z,.+4*       !WATCH THIS FOR GOD'S SAKE
13C7 C603                      ADD#3
13C9 CDB70E                    CALL CHOUT
13CC DD6E00                    L<-0(IX)
13CF DD6601                    H<-1(IX)
13D2 CDC612CDA80E202000        CALL PR16*CALL PRINT*.ASCIZ%  *
13DB 210F1334CB5628D7          HL<-#SYMNUM*INC (HL)*TEST 2,(HL)*BR Z,SYMRET
13E3 CDCC0EAF320F13            CALL CRLF*CLA*SYMNUM<-A
13EA C1DDE1C9        SYMRET:   POP BC,IX*RET
13EE E5C5            ERRPRN:   PUSH HL,BC*!     PRINT ANY ERRORS FOUND
13F0 21E212060A                HL<-#<ONEERR-ERRNUM>+1*B<-#ERRNUM
13F5 7EB7C40014 2310F8         A<-(HL)*TSTA*CALL NZ,ERROP*INC HL*DBNZ .
13FD E1C1C9                    POP HL,BC*RET
1400 CDA80E4552524F5220 ERROP: CALL PRINT*.ASCIZ%ERROR *
140A 78CDC612CDCC0E            A<-B*CALL PR2*CALL CRLF
1411 C9                        RET
1412 F5C5            BLANK:    PUSH AF,BC*!     PRODUCE LEADER OR TRAILER
1414 AF0680                    CLA*B<-#200
1417 CDB70E10FB                CALL CHOUT*DBNZ .
141C C1F1C9                    POP BC,AF*RET
141F                 ONETAB:
141F 5245540000 00C900         .WORD 'R,'E,'T,0,0,0,311,0
1427 4441410000002700          .WORD 'D,'A,'A,0,0,0,47,0
142F 545354410000B700          .WORD 'T,'S,'T,'A,0,0,267,0
1437 434C41000000AF00          .WORD 'C,'L,'A,0,0,0,257,0
143F 455858000000D900          .WORD 'E,'X,'X,0,0,0,331,0
1447 4C4449000000EDA0          .WORD 'L,'D,'I,0,0,0,355,240
144F 4C444952000 0EDB0         .WORD 'L,'D,'I,'R,0,0,355,260
1457 4C4444000000EDA8          .WORD 'L,'D,'D,0,0,0,355,250
145F 4C44445200 00EDB8         .WORD 'L,'D,'D,'R,0,0,355,270
1467 435049000000EDA1          .WORD 'C,'P,'I,0,0,0,355,241
146F 4350495200 00EDB1         .WORD 'C,'P,'I,'R,0,0,355,261
1477 435044000000EDA9          .WORD 'C,'P,'D,0,0,0,355,251
147F 4350445200 00EDB9         .WORD 'C,'P,'D,'R,0,0,355,271
1487 434D410000002F00          .WORD 'C,'M,'A,0,0,0,57,0
148F 4E4547000000ED44          .WORD 'N,'E,'G,0,0,0,355,104
1497 4343460000003F00          .WORD 'C,'C,'F,0,0,0,77,0
149F 5343460000003700          .WORD 'S,'C,'F,0,0,0,67,0
14A7 4E4F500000000000          .WORD 'N,'O,'P,0,0,0,0,0
14AF 48414C5400007600          .WORD 'H,'A,'L,'T,0,0,166,0
14B7 444900000000F300          .WORD 'D,'I,0,0,0,0,363,0
14BF 454900000000FB00          .WORD 'E,'I,0,0,0,0,373,0
14C7 524C434100000700          .WORD 'R,'L,'C,'A,0,0,7,0
14CF 524C410000001700          .WORD 'R,'L,'A,0,0,0,27,0
14D7 5252434100000F00          .WORD 'R,'R,'C,'A,0,0,17,0
14DF 5252410000001F00          .WORD 'R,'R,'A,0,0,0,37,0
14E7 524C44000000ED6F          .WORD 'R,'L,'D,0,0,0,355,157
14EF 525244000000ED67          .WORD 'R,'R,'D,0,0,0,355,147
14F7 524554490000ED4D          .WORD 'R,'E,'T,'I,0,0,355,115
14FF 5245544E0000ED45          .WORD 'R,'E,'T,'N,0,0,355,105
1507 494E49000000EDA2          .WORD 'I,'N,'I,0,0,0,355,242
150F 494E49520000EDB2          .WORD 'I,'N,'I,'R,0,0,355,262
1517 494E44000000EDAA          .WORD 'I,'N,'D,0,0,0,355,252
151F 494E44520000EDBA          .WORD 'I,'N,'D,'R,0,0,355,272
1527 4F5554490000EDA3          .WORD 'O,'U,'T,'I,0,0,355,243
152F 4F5449520000EDB3          .WORD 'O,'T,'I,'R,0,0,355,263
1537 4F5554440000EDAB          .WORD 'O,'U,'T,'D,0,0,355,253

153F 4F5444520000EDBB          .WORD 'O,'T,'D,'R,0,0,355,273
1547 494D30000000ED46          .WORD 'I,'M,'0,0,0,0,355,106
154F 494D31000000ED56          .WORD 'I,'M,'1,0,0,0,355,126
1557 494D32000000ED5E          .WORD 'I,'M,'2,0,0,0,355,136
155F 00                        .WORD 0
1560 2E00000000800000 DOTRFS:  .WORD '.,0,0,0,0,200,0,0
1568                 CURADD:   .BLKA
156A BC15            SYMNEX:   .ADDR SYMSRT
156C                 TBUFF:    .BLKW 100
15AC                 ATBLCK:   .BLKW 10
15B4                           !IF IT OVERFLOWS,ONLY SYMBOL TABLE LOST
15B4                 SYMTAB:   .BLKW 6
15BA                 OURDOT:   .BLKA
15BC 00              SYMSRT:   .WORD 0*!      SYMBOL TABLE FROM HERE ON UPWARDS
15BD                           RUBOUT=177*SEMICO=';*CR=15*LF=12
15BD                           STACK=400
15BD                           TTYSTA=335*TTYTR=334*TTYCNT=336
15BD                           COLON=':*CLOBRK='!
15BD                           TAB='   *SPACE=40
15BD                           TERMIN=1*ALPH=2*LABL=20
15BD                           NUM=10*MATH=4*ALFNUM=100
15BD                           ERRNUM=12*CWIDTH=5
15BD                           .END BEGIN

NO. OF DETECTED ERRORS:
```

PCW *May we remind contributors that listings must be bold and clear. We apologise for the quality of reproduction of this series. We cannot emphasise too strongly that listings which are not of good quality cannot be considered for publication.* **PCW.**

THE BYTE COSERVER

Graham Trott



6800 MICRO-ASSEMBLER

Any serious user of a microprocessor is going to get involved in assembly-language programming sooner or later. BASIC is fine for computer-type applications, but operations on a bit level, controlling relays, or squeezing the utmost in performance out of the machine require the user to understand and control the detailed operation of the processor in a way that high-level languages are either not suited to or are designed to make unnecessary (as long as you stick to number-crunching). Many users, of course, either cannot afford the luxury of sufficient memory to run a decent version of BASIC, or are perhaps writing programs to run in ROM, which effectively bars the use of interpreters (although not compilers).

An assembler, then, is high on the software shopping list, unless you are the sort of freak that can think in two's complement hexadecimal and remember a couple of hundred op-codes. Most assemblers either require around 8K of RAM to run in or need a separate editor to create the source program. In either case you spend most of your time loading programs from tape; and if that's not enough, you often require two cassette drives, at least one with remote control facilities.

There are many occasions when the program you wish to try out is only a dozen or so lines long. A conventional assembler is inconvenient to use under these circumstances, and indeed you may be trying to patch a program that occupies the *same* area as your assembler. The micro-assembler that I am about to describe is written for the 6800. It is designed to run in ROM, and is therefore always available. It requires the use of no editor or mass-storage device, since its function is to translate typed mnemonics directly into machine code, for immediate execution or for later use.

Upon starting up, the micro-assembler requests the address at which the assembled code is to be saved. It then accepts three characters from the terminal and checks that they constitute a valid mnemonic. All of the 6800 mnemonics are catered for, together with a few useful extras, viz:—

BHS (Branch if Higher or Same) = BCC
BLO (Branch if LOwer) = BCS
SK1 (Skip one byte)
SK2 (Skip two bytes)

NOTE: the last two generate $81 (CMPA) and $8C (CPX) respectively. Do not use them unless you understand the implications as regards the condition code register.

The assembler pseudo-instructions FCC and FCB allow the direct insertion of text and hex digits respectively into memory. The former is terminated by Control D ($04), which is saved as the last character of the text, and the latter is terminated by any non-hex character.

Once the mnemonic has been accepted, the assembler outputs a space. At this point, some instructions are complete, e.g. INX, CLC, TBA, etc. In that case, the corresponding machine instruction is displayed and written to memory. (The micro-assembler will always check that there is RAM to accept object code.) If the instruction is not complete, further input is required. This may be simply A or B, in the case of accumulator inherent addressing. Multiple-byte instructions must indicate the mode of addressing, i.e. Immediate (I), Direct (D), Indexed (X) or Extended (E). Relative (branch) instructions require no identifier. Lastly the operand must be supplied, in hexadecimal characters representing either 8 or 16 bits (the assembler will always know which). A typical sequence might be:—

```
ADDRESS – 0100
0100  – LDA A I 01              86 01
0102  – ABA                     1B
0103  – STA A I*** ERROR ***    (store immediate is illegal)
0103  – STA A D 31              97 31
0105  – LDX E 013F              FE 013F
0108  – TST X 00                6D 00
010A  – BRA 0102                20 F6
010C  – FCC THIS IS TEXT DIRECTLY INSERTED INTO
           MEMORY.
0137  – FCB 04 FF 6A B8 33 89 <CR>
013D  – <CR>
ADDRESS – E000
E000  – NOP                     *** ERROR *** (no RAM)
E000  – <CR>
ADDRESS – <CR>
*   (Back in MIKBUG)
```

Note in particular the use of the BRA instruction. The operand is calculated by the assembler and an error will result if the range is too great. Note also that a carriage-return is used to exit the assembler or to start assembly at a new address. In the above example, all spaces were inserted by the assembler – all the user types is the mnemonics and the operands (in hex only).

The listing shows the assembler to have its origin at $1000, which will locate it at the top of an 8K system, but since it is fully relocatable, it may be run at any convenient address. I am prepared to make available copies of the micro-assembler, either on CUTS cassette or in a 2708 EPROM. In the former case I will also include a routine that will enable the user to move the assembler to any desired RAM position. Anyone interested should contact me at the following address: G. J. Trott, 99 Mill Lane, Felixstowe, Suffolk IP11 8LN.

```
6800 MICRO - ASSEMBLER                    SSB MNEMONIC ASSEMBLER

                  * 6800 MICRO - ASSEMBLER IS
                  *      RELOCATABLE
                  *      ROM - ABLE
                  *      INTERRUPTABLE
                  * AND OCCUPIES LESS THAN 1024 BYTES
                  *
                  * IT IS ASSEMBLED TO LOAD AT THE TOP
                  * OF AN 8-K SYSTEM (STARTING AT $1C00)
                  * BUT CAN BE MOVED AND RUN ANYWHERE.
                  *
                  * THE FOLLOWING MIKBUG ROUTINES
                  * MUST BE PRESENT.  ANY MIKBUG -
                  * COMPATIBLE MONITOR WILL BE SUITABLE.

E1AC              INCH   EQU    $E1AC
E1D1              OUTCH  EQU    $E1D1
E07E              PDATA1 EQU    $E07E
E047              BADDR  EQU    $E047
E0BF              OUT2H  EQU    $E0BF
E0C8              OUT4HS EQU    $E0C8
E0CA              OUT2HS EQU    $E0CA
E0CC              OUTS   EQU    $E0CC

004E              NCODES EQU    TABEND-TABLE/4

1C00                     ORG    $1C00

1C00 8E A0 42     ASSEM  LDS    #STACK
1C03 8D 0F               BSR    PRTAD
1C05 0D                  FCB    13,10,0,0
1C09 41                  FCC    'ADDRESS - '
1C13 04                  FCB    4
1C14 30           PRTAD  TSX
1C15 EE 00               LDX    X           POINT TO STRING
1C17 31                  INS
1C18 31                  INS
1C19 BD E0 7E            JSR    PDATA1      "ADDRESS - "
1C1C BD E0 47            JSR    BADDR       INPUT ADDRESS
1C1F 8E A0 42     NEWLIN LDS    #STACK
1C22 8D 05               BSR    PCRLF
1C24 0D                  FCB    13,10,0,0,4
1C29 30           PCRLF  TSX
1C2A EE 00               LDX    X           POINT TO STRING
1C2C 31                  INS
1C2D 31                  INS
1C2E BD E0 7E            JSR    PDATA1      CR - LF
1C31 86 0F               LDA A  #15
1C33 B7 A0 4E            STA A  TAB
1C36 B7 A0 4F            STA A  OPCODE      FLAG FOR ERROR ROUTINE
1C39 CE A0 0C            LDX    #XHI

1C3C BD E0 C8            JSR    OUT4HS      PRINT THE ADDRESS
1C3F 86 2D               LDA A  #'-
1C41 BD E1 D1            JSR    OUTCH       PRINT'-'
1C44 BD E0 CC            JSR    OUTS        PRINT A SPACE

                  * INPUT LOOP

1C47 CE A0 4A            LDX    #SBUF       POINT TO SOURCE BUFFER
1C4A BD E1 AC     CHRIN  JSR    INCH
1C4D 81 0D               CMP A  #$D         RESTART?
1C4F 27 AF               BEQ    ASSEM
1C51 81 1B               CMP A  #$1B        ESCAPE?
1C53 27 CA        NEW3B  BEQ    NEWLIN      RESTART THIS LINE
1C55 A7 00               STA A  X           SAVE THE CHARACTER
1C57 08                  INX
1C58 8C A0 4D            CPX    #SBUF+3
1C5B 26 ED               BNE    CHRIN       GET 3 CHARACTERS
1C5D BD E0 CC            JSR    OUTS        PRINT A SPACE

                  * MATCH THE MNEMONIC AGAINST THE TABLE

1C60 7F A0 4D            CLR    OPCSER      CLEAR POSITION INDICATOR
1C63 86 1E               LDA A  #TABLE/256
1C65 C6 7A               LDA B  #TABLE
1C67 8D 45               BSR    OFFCAL      RELOCATOR TO GET TABLE ADDRESS
1C69 FF A0 54     MATCH  STX    TEMP1
1C6C C6 03               LDA B  #3
1C6E CE A0 4A            LDX    #SBUF
1C71 A6 00        MAT2   LDA A  X
1C73 08                  INX
1C74 FF A0 56            STX    TEMP2
1C77 FE A0 54            LDX    TEMP1
1C7A A1 00               CMP A  X           TEST THE CHARACTER
1C7C 26 44               BNE    NEXT
1C7E 08                  INX
1C7F FF A0 54            STX    TEMP1       POINT TO THE NEXT
1C82 FE A0 56            LDX    TEMP2
1C85 5A                  DEC B
1C86 26 E9               BNE    MAT2        REPEAT 'TILL 3 DONE
1C88 FE A0 54            LDX    TEMP1
1C8B E6 00               LDA B  X
1C8D F7 A0 4F            STA B  OPCODE      SAVE THE OPCODE

                  * MNEMONIC FOUND. NOW JUMP TO
                  * RELEVANT SECTION OF PROGRAM.

1C90 37                  PSH B
1C91 86 1F               LDA A  #JTABLE-3/256
1C93 C6 AE               LDA B  #JTABLE-3
1C95 8D 17               BSR    OFFCAL      GET THE TABLE ADDRESS
1C97 B6 A0 4D            LDA A  OPCSER      TYPE OF OPCODE

1C9A 4C                  INC A              ADJUST IT
1C9B 08           JTABL  INX
1C9C 08                  INX
1C9D 08                  INX
1C9E A1 00               CMP A  X           FIND THE INSTRUCTION TYPE
1CA0 23 F9               BLS    JTABL
1CA2 A6 01               LDA A  1,X
1CA4 E6 02               LDA B  2,X
1CA6 8D 06               BSR    OFFCAL      FIND THE SERVICE ROUTINE
1CA8 33                  PUL B
1CA9 6E 00               JMP    X           ...AND JUMP TO IT

1CAB 4F           NEW3A  CLR A
1CAC 20 A5               BRA    NEW3B
```

```
                  * TO ENABLE THE ASSEMBLER TO BE RELOCATABLE,
                  * THIS SUBROUTINE ADJUSTS TABLE ADDRESSES.

1CAE 8D 00        OFFCAL BSR    GETTBL
1CB0 30           GETTBL TSX
1CB1 C0 B0               SUB B  #GETTBL
1CB3 82 1C               SBC A  #GETTBL/256 CALCULATE THE OFFSET
1CB5 EB 01               ADD B  1,X
1CB7 A9 00               ADC A  X           ADD IT TO CURRENT POSITION
1CB9 A7 00               STA A  X
1CBB E7 01               STA B  1,X         THIS IS THE ACTUAL ADDRESS
1CBD EE 00               LDX    X           PICK IT UP IN X
1CBF 31                  INS
1CC0 31                  INS
1CC1 39                  RTS

                  * NO MATCH - STEP ON TO NEXT OPCODE

1CC2 08           NEXT   INX
1CC3 5A                  DEC B
1CC4 26 FC               BNE    NEXT        STEP PAST THIS ONE
1CC6 08                  INX                SKIP THE OPCODE
1CC7 7C A0 4D            INC    OPCSER
1CCA B6 A0 4D            LDA A  OPCSER
1CCD 81 4E               CMP A  #NCODES     CHECK IF ANY MORE
1CCF 26 98               BNE    MATCH

1CD1 7D A0 4F     ERROR  TST    OPCODE
1CD4 27 17               BEQ    NEW3        (SEE HEXIN)
1CD6 8D 0D               BSR    PERROR
1CD8 2A                  FCC    '** ERROR **'
1CE3 07                  FCB    7,4
1CE5 30           PERROR TSX
1CE6 EE 00               LDX    X           POINT TO STRING
1CE8 31                  INS
1CE9 31                  INS

1CEA BD E0 7E            JSR    PDATA1      "** ERROR **"
1CED 20 BC        NEW3   BRA    NEW3A       NEXT LINE

                  * SERVICE ROUTINES FOR THE DIFFERENT INSTRUCTION TYPES

                  * ASCII INPUT (FCC). STOP WHEN CONTROL D ENTERED.

1CEF FE A0 0C     ASCII  LDX    XHI
1CF2 BD E1 AC     ASC2   JSR    INCH        GET A CHARACTER
1CF5 A7 00               STA A  X           SAVE IT
1CF7 08                  INX
1CF8 09                  DEX
1CF9 A1 00               CMP A  X           DID IT SAVE?
1CFB 26 D4               BNE    ERROR
1CFD 08                  INX
1CFE FF A0 0C            STX    XHI
1D01 81 04               CMP A  #4          END OF STRING?
1D03 26 ED               BNE    ASC2        CONTINUE IF NOT
1D05 20 A4        NEW2   BRA    NEW3A       NEXT LINE

                  * NUMERIC INPUT (FCB)

1D07 FE A0 0C     HEXIN  LDX    XHI
1D0A 7F A0 4F            CLR    OPCODE      DON'T GENERATE ERROR MESSAGE
1D0D 8D 74        HEX2   BSR    INB2        GET A BYTE
1D0F A7 00               STA A  X
1D11 08                  INX
1D12 09                  DEX
1D13 A1 00               CMP A  X
1D15 26 BA               BNE    ERROR
1D17 08                  INX
1D18 FF A0 0C            STX    XHI
1D1B BD E0 CC            JSR    OUTS
1D1E 20 ED               BRA    HEX2        NEXT BYTE

                  * BRANCH OPERATIONS

1D20 8D 6A        BRANCH BSR    OPER11
1D22 8D 6A               BSR    OPER21
1D24 7C A0 4E            INC    TAB         NO MODE REQUIRED HERE
1D27 F6 A0 50            LDA B  OPND1       LOAD ADDRESS
1D2A 80 02               SUB A  #2
1D2C C2 00               SBC B  #0
1D2E B0 A0 0D            SUB A  XHI+1
1D31 F2 A0 0C            SBC B  XHI
1D34 B7 A0 50            STA A  OPND1       SAVE OFFSET
1D37 49                  ROL A              NOW CHECK VALIDITY
1D38 59                  ROL B
1D39 27 60               BEQ    TWOBYT      ERROR IF A -VE, B=0, C=0
1D3B 53                  COM B
1D3C 26 93               BNE    ERROR       OR IF A +VE, B=FF, C=1

1D3E 20 5B               BRA    TWOBYT

1D40 20 8F        ERR6   BRA    ERROR
1D42 20 C1        NEW5   BRA    NEW2

                  * ACCUMULATOR OPERATIONS

1D44 BD E1 AC     ACCUM  JSR    INCH        GET THE MODIFIER
1D47 81 41               CMP A  #'A
1D49 27 06               BEQ    ACC2
1D4B 81 42               CMP A  #'B
1D4D 26 3B               BNE    ERR2        ERROR - INVALID CHARACTER
1D4F CB 40               ADD B  #$40        ACCB OPERATION
1D51 BD E0 CC     ACC2   JSR    OUTS        PRINT A SPACE
1D54 7A A0 4E            DEC    TAB
1D57 8D 7A               BSR    MODE        FIND THE ADDRESSING MODE
1D59 7A A0 4E            DEC    TAB
1D5C 4D                  TST A
1D5D 26 08               BNE    NOZERO      NOT IMMEDIATE ADDRESSING
1D5F B6 A0 4F            LDA A  OPCODE
1D62 81 87               CMP A  #$87        CHECK FOR STORE IMMEDIATE
1D64 27 DA               BEQ    ERR6        IT CAN'T BE DONE
1D66 4F                  CLR A
1D67 8D 36        NOZERO BSR    ADJUS2
1D69 81 30               CMP A  #$30        EXTENDED?
1D6B 27 34               BEQ    THREBT      THREE BYTES
1D6D 20 2C               BRA    TWOBYT      TWO BYTES

                  * PUSH AND PULL

1D6F BD E1 AC     PULPSH JSR    INCH        GET THE MODIFIER
1D72 81 41               CMP A  #'A
```

```
1D74 27 44           BEQ    MEM3
1D76 81 42           CMP A  #'B
1D78 26 10           BNE    ERR2      ERROR - INVALID MNEMONIC
1D7A 5C              INC B            B ACCUMULATOR ADDRESSED
1D7B 20 3D           BRA    MEM3

              * JMP, JSR

1D7D 8D 54    JMPS   BSR    MODE      GET THE MODE OF ADDRESSING
1D7F 81 10    JUM2   CMP A  #$10      ONLY INDEXED OR EXTENDED
1D81 20 05           BRA    ST16A

1D83 20 4C    INB2   BRA    INB3

              * STX, STS

1D85 8D 4C    ST16   BSR    MODE      GET THE MODE OF ADDRESSING
1D87 4D              TST A            CAN'T BE IMMEDIATE
1D88 22 08    ST16A  BHI    EXT16A

1D8A 20 B4    ERR2   BRA    ERR6

1D8C 20 63    OPER11 BRA    OPER12
1D8E 20 63    OPER21 BRA    OPER22

              * CPX, LDX, LDS

1D90 8D 41    EXT16  BSR    MODE      FIND THE MODE OF ADDRESSING
1D92 8D 65    EXT16A BSR    ADJUST    ADJUST THE OPCODE
1D94 4D              TST A            TEST FOR IMMEDIATE
1D95 27 0A           BEQ    THREBT
1D97 81 30           CMP A  #$30      OR EXTENDED
1D99 27 06           BEQ    THREBT
1D9B 8D 7B    TWOBYT BSR    SAV21
1D9D 20 A3    NEW4   BRA    NEW5      NEXT LINE

1D9F 20 58    ADJUS2 BRA    ADJUST

1DA1 8D 50    THREBT BSR    OPER22    GET ANOTHER OPERAND
1DA3 8D 73           BSR    SAV21
1DA5 8D 73           BSR    SAV31
1DA7 20 F4           BRA    NEW4      NEXT LINE

              * MEMORY OPERATIONS

1DA9 BD E1 AC MEM    JSR    INCH      GET THE MODIFIER
1DAC 81 41           CMP A  #'A
1DAE 27 0A           BEQ    MEM3
1DB0 81 42           CMP A  #'B
1DB2 27 04           BEQ    MEM2
1DB4 8D 20           BSR    MOD2      IF NOT ACCUMULATOR A,B
1DB6 20 C7           BRA    JUM2      THEN DO AS FOR JMP, JSR
1DB8 CB 10    MEM2   ADD B  #$10
1DBA F7 A0 4F MEM3   STA B  OPCODE
1DBD 8D 36           BSR    TTAB2     TAB ACROSS PAGE
1DBF CE A0 4F        LDX    #OPCODE
1DC2 BD E0 BF        JSR    OUT2H     PRINT THE OPCODE
1DC5 F6 A0 4F        LDA B  OPCODE
1DC8 8D 52           BSR    SAV3AA    AND SAVE IT
1DCA 20 D1           BRA    NEW4      NEXT LINE

              * INHERENT INSTRUCTIONS

1DCC 7C A0 4E INHRNT INC    TAB       NO MODE IDENTIFIER HERE
1DCF 20 E9           BRA    MEM3

1DD1 20 57    INB3   BRA    INBYTE

              * SOME SUBROUTINES

              * GET THE MODE OF ADDRESSING (I,D,X OR E)

1DD3 BD E1 AC MODE   JSR    INCH
1DD6 81 49    MOD2   CMP A  #'I       IMMEDIATE?
1DD8 26 02           BNE    MOD3
1DDA 4F              CLR A
1DDB 39              RTS
1DDC 81 44    MOD3   CMP A  #'D       DIRECT?
1DDE 26 03           BNE    MOD4
1DE0 86 10           LDA A  #$10
1DE2 39              RTS
1DE3 81 58    MOD4   CMP A  #'X       INDEXED?
1DE5 26 03           BNE    MOD5
1DE7 86 20           LDA A  #$20
1DE9 39              RTS
1DEA 81 45    MOD5   CMP A  #'E       EXTENDED?
1DEC 26 9C           BNE    ERR2      ERROR IF NOT
1DEE 86 30           LDA A  #$30
1DF0 39              RTS

1DF1 20 44    OPER12 BRA    OPER1
1DF3 20 48    OPER22 BRA    OPER2
1DF5 20 4C    TTAB2  BRA    TTABB
1DF7 20 91    ERR5   BRA    ERR2

              * ADJUST THE OPCODE

1DF9 36       ADJUST PSH A
1DFA 1B              ABA
1DFB B7 A0 4F        STA A  OPCODE    SAVE IT
1DFE 8D 37           BSR    OPER1     GET AN OPERAND
1E00 32              PUL A
1E01 39              RTS

              * INPUT A HEX CHARACTER

1E02 BD E1 AC HEXCHR JSR    INCH
1E05 80 30           SUB A  #'0
1E07 2B EE           BMI    ERR5      ERROR NON-HEX CHARACTER
1E09 81 09           CMP A  #9
1E0B 2F 0A           BLE    OKRET
1E0D 81 11           CMP A  #$11      "A"
1E0F 25 E6           BLO    ERR5      ERROR
1E11 81 16           CMP A  #$16      "F"
1E13 22 E2           BHI    ERR5      ERROR
1E15 80 07           SUB A  #7
1E17 39       OKRET  RTS

1E18 20 32    SAV21  BRA    SAVE2
1E1A 20 45    SAV31  BRA    SAVE3
1E1C 20 4C    SAV3AA BRA    SAV3A

              * PRINT A SPACE THEN DECREMENT TAB THRICE

1E1E BD E0 CC PRSP   JSR    OUTS
1E21 7A A0 4E        DEC    TAB
1E24 7A A0 4E TABADJ DEC    TAB
1E27 7A A0 4E        DEC    TAB

              * GET TWO HEX CHARACTERS AS ONE BYTE

1E2A 8D D6    INBYTE BSR    HEXCHR
1E2C 48              ASL A
1E2D 48              ASL A
1E2E 48              ASL A
1E2F 48              ASL A
1E30 16              TAB
1E31 8D CF           BSR    HEXCHR
1E33 1B              ABA
1E34 39              RTS

1E35 20 C0    ERR4   BRA    ERR5

              * GET AN OPERAND

1E37 8D E5    OPER1  BSR    PRSP
1E39 B7 A0 50        STA A  OPND1
1E3C 39              RTS

              * GET A SECOND OPERAND

1E3D 8D E5    OPER2  BSR    TABADJ
1E3F B7 A0 51        STA A  OPND2
1E42 39              RTS

              * TAB ACROSS PAGE

1E43 BD E0 CC TTABB  JSR    OUTS      PRINT SPACE
1E46 7A A0 4E        DEC    TAB
1E49 26 F8           BNE    TTABB
1E4B 39              RTS

              * PRINT AND SAVE 2-BYTE INSTRUCTIONS

1E4C 8D F5    SAVE2  BSR    TTABB
1E4E CE A0 4F        LDX    #OPCODE
1E51 BD E0 CA        JSR    OUT2HS    PRINT OPCODE
1E54 BD E0 BF        JSR    OUT2H     PRINT OPERAND
1E57 F6 A0 4F        LDA B  OPCODE
1E5A 8D 0E           BSR    SAV3A     STORE OPCODE
1E5C F6 A0 50        LDA B  OPND1
1E5F 20 0C           BRA    SAV3B     STORE OPERAND
              * PRINT AND SAVE 3RD BYTE

1E61 CE A0 51 SAVE3  LDX    #OPND2
1E64 BD E0 BF        JSR    OUT2H     PRINT OPERAND
1E67 F6 A0 51        LDA B  OPND2
1E6A FE A0 0C SAV3A  LDX    XHI
1E6D E7 00    SAV3B  STA B  X         SAVE OPERAND
1E6F 08              INX
1E70 09              DEX
1E71 E1 00           CMP B  X         DID IT SAVE?
1E73 26 C0           BNE    ERR4      ERROR IF NOT
1E75 08              INX
1E76 FF A0 0C        STX    XHI
1E79 39              RTS

              * MNEMONIC LOOK-UP TABLE

1E7A          TABLE  EQU    *
1E7A 4E              FCB    'N,'O,'P,$01,'T,'A,'P,$06
1E82 54              FCB    'T,'P,'A,$07,'I,'N,'X,$08
1E8A 44              FCB    'D,'E,'X,$09,'C,'L,'V,$0A
1E92 53              FCB    'S,'E,'V,$0B,'C,'L,'C,$0C
1E9A 53              FCB    'S,'E,'C,$0D,'C,'L,'I,$0E
1EA2 53              FCB    'S,'E,'I,$0F,'S,'B,'A,$10
1EAA 43              FCB    'C,'B,'A,$11,'T,'A,'B,$16
1EB2 54              FCB    'T,'B,'A,$17,'D,'A,'A,$19
1EBA 41              FCB    'A,'B,'A,$1B,'T,'S,'X,$30
1EC2 49              FCB    'I,'N,'S,$31,'D,'E,'S,$34
1ECA 54              FCB    'T,'X,'S,$35,'R,'T,'S,$39
1ED2 52              FCB    'R,'T,'I,$3B,'W,'A,'I,$3E
1EDA 53              FCB    'S,'W,'I,$3F
1EDE 53              FCB    'S,'K,'1,$81,'S,'K,'2,$8C
1EE6          EINH   EQU    *
1EE6 42              FCB    'B,'R,'A,$20,'B,'H,'I,$22
1EEE 42              FCB    'B,'L,'S,$23,'B,'C,'C,$24
1EF6 42              FCB    'B,'H,'S,$24,'B,'C,'S,$25
1EFE 42              FCB    'B,'L,'O,$25,'B,'N,'E,$26
1F06 42              FCB    'B,'E,'Q,$27,'B,'V,'C,$28
1F0E 42              FCB    'B,'V,'S,$29,'B,'P,'L,$2A
1F16 42              FCB    'B,'M,'I,$2B,'B,'G,'E,$2C
1F1E 42              FCB    'B,'L,'T,$2D,'B,'G,'T,$2E
1F26 42              FCB    'B,'L,'E,$2F,'B,'S,'R,$8D
1F2E          EREL   EQU    *
1F2E 53              FCB    'S,'U,'B,$80,'C,'M,'P,$81
1F36 53              FCB    'S,'B,'C,$82,'A,'N,'D,$84
1F3E 42              FCB    'B,'I,'T,$85,'L,'D,'A,$86
1F46 53              FCB    'S,'T,'A,$87,'E,'O,'R,$88
1F4E 41              FCB    'A,'D,'C,$89,'O,'R,'A,$8A
1F56 41              FCB    'A,'D,'D,$8B
1F5A          EACC   EQU    *
1F5A 4E              FCB    'N,'E,'G,$40,'C,'O,'M,$43
1F62 4C              FCB    'L,'S,'R,$44,'R,'O,'R,$46
1F6A 41              FCB    'A,'S,'R,$47,'A,'S,'L,$48
1F72 52              FCB    'R,'O,'L,$49,'D,'E,'C,$4A
1F7A 49              FCB    'I,'N,'C,$4C,'T,'S,'T,$4D
1F82 43              FCB    'C,'L,'R,$4F
1F86          EMEM   EQU    *
1F86 50              FCB    'P,'U,'L,$32,'P,'S,'H,$36
1F8E          EPP    EQU    *
1F8E 43              FCB    'C,'P,'X,$8C,'L,'D,'S,$8E
1F96 4C              FCB    'L,'D,'X,$CE
1F9A          E16    EQU    *
1F9A 53              FCB    'S,'T,'S,$8F,'S,'T,'X,$CF
1FA2          ES16   EQU    *
1FA2 4A              FCB    'J,'S,'R,$8D,'J,'M,'P,$4E
1FAA          EJMP   EQU    *
1FAA 46              FCB    'F,'C,'C,$01
1FAE          EFCC   EQU    *
1FAE 46              FCB    'F,'C,'B
1FB2          TABEND EQU    *+1
```

Cont. p52

```
                    * TRANSFER TABLE (ADDRESSES OF SERVICE ROUTINES)

1FB1 4D             JTABLE FCB    EFCC-TABLE/4
1FB2 1D 07                 FDB    HEXIN
1FB4 4C                    FCB    EJMP-TABLE/4
1FB5 1C EF                 FDB    ASCII
1FB7 4A                    FCB    ES16-TABLE/4
1FB8 1D 7D                 FDB    JMPS
1FBA 48                    FCB    E16-TABLE/4
1FBB 1D 85                 FDB    ST16
1FBD 45                    FCB    EPP-TABLE/4
1FBE 1D 90                 FDB    EXT16
1FC0 43                    FCB    EMEM-TABLE/4
1FC1 1D 6F                 FDB    PULPSH
1FC3 38                    FCB    EACC-TABLE/4
1FC4 1D A9                 FDB    MEM
1FC6 2D                    FCB    EREL-TABLE/4
1FC7 1D 44                 FDB    ACCUM
1FC9 1B                    FCB    EINH-TABLE/4
1FCA 1D 20                 FDB    BRANCH
1FCC 00                    FCB    0
1FCD 1D CC                 FDB    INHRNT

                    * RAM STORAGE LOCATIONS

A00C                XHI    EQU    $A00C

A042                       ORG    $A042

A042                STACK  RMB    8
A04A                SBUF   RMB    3

A04D                OPCSER RMB    1     A051         OPND2  RMB   1
A04E                TAB    RMB    1     A052         MEMADR RMB   2
A04F                OPCODE RMB    1     A054         TEMP1  RMB   2
A050                OPND1  RMB    1     A056         TEMP2  RMB   2

                                                            END
                                              NO ERROR(S) DETECTED

 SYMBOL TABLE:
ACC2    1D51    ACCUM   1D44    ADJUS2  1D9F    ADJUST  1DF9
ASC2    1CF2    ASCII   1CEF    ASSEM   1C00    BADDR   E047
BRANCH  1D20    CHRIN   1C4A    E16     1F9A    EACC    1F5A
EFCC    1FAE    EINH    1EE6    EJMP    1FAA    EMEM    1F86
EPP     1F8E    EREL    1F2E    ERR2    1D8A    ERR4    1E35
ERR5    1DF7    ERR6    1D40    ERROR   1CD1    ES16    1FA2
EXT16   1D90    EXT16A  1D92    GETTBL  1CB0    HEX2    1D0D
HEXCHR  1E02    HEXIN   1D07    INB2    1D83    INB3    1DD1
INBYTE  1E2A    INCH    E1AC    INHRNT  1DCC    JMPS    1D7D
JTABL   1C9B    JTABLE  1FB1    JUM2    1D7F    MAT2    1C71
MATCH   1C69    MEM     1DA9    MEM2    1DB8    MEM3    1DBA
MEMADR  A052    MOD2    1DD6    MOD3    1DDC    MOD4    1DE3
MOD5    1DEA    MODE    1DD3    NCODES  004E    NEW2    1D05
NEW3    1CED    NEW3A   1CAB    NEW3B   1C53    NEW4    1D9D
NEW5    1D42    NEWLIN  1C1F    NEXT    1CC2    NOZERO  1D67
OFFCAL  1CAE    OKRET   1E17    OPCODE  A04F    OPCSER  A04D
OPER1   1E37    OPER11  1D8C    OPER12  1DF1    OPER2   1E3D
OPER21  1D8E    OPER22  1DF3    OPND1   A050    OPND2   A051
OUT2H   E0BF    OUT2HS  E0CA    OUT4HS  E0C8    OUTCH   E1D1
OUTS    E0CC    PCRLF   1C29    PDATA1  E07E    PERROR  1CE5
PRSP    1E1E    PRTAD   1C14    PULPSH  1D6F    SAV21   1E18
SAV31   1E1A    SAV3A   1E6A    SAV3AA  1E1C    SAV3B   1E6D
SAVE2   1E4C    SAVE3   1E61    SBUF    A04A    ST16    1D85
ST16A   1D88    STACK   A042    TAB     A04E    TABADJ  1E24
TABEND  1FB2    TABLE   1E7A    TEMP1   A054    TEMP2   A056
THREBT  1DA1    TTAB2   1DF5    TTABB   1E43    TWOBYT  1D9B
XHI     A00C
```

PERSONAL COMPUTER WORLD

Some Evaluations or User Reports you can look forward to

The HORIZON
The EQUINOX 300
The SORCERER
The OSI SUPERBOARD II (and others)
The SYM by SYNERTEK
The PET (Re-evaluation)
The MSI
The ERA Microtutor
The LIMROSE Microtutor
The VECTOR MZ
The IMSAI
The SWTPC 6800
And more . . . .

The Faces Behind the Places

Some reports on computer suppliers you can look forward to

The BYTE SHOP
MICRODIGITAL OF LIVERPOOL
BELVEDERE
JADE
COMMODORE BUSINESS SYSTEMS
NEWBEAR
SEED
PERSONAL COMPUTERS LTD
L.P. ENTERPRISES
NASCOM MICROCOMPUTERS
RESEARCH MACHINES LTD
SINTROM MICROSHOP
COMPUTER WORKSHOP
COMP COMPUTER COMPONENTS
TANDY

STATPACK

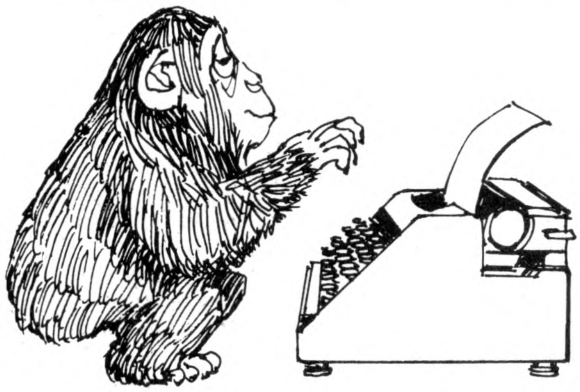
Part 5

Colin Chatfield

```
0005 REM STAT5
0020 INPUT "     ENTER OUTPUT PORT # ",Z9
0080 GOSUB 9380: LINE=132
0100 ? TAB(20);"STAT5 FOR PROGRESSIVE AVERAGE":?
1150 GOSUB 9600
1160 ? "YOUR ARRAY IS SET AT ";A;"x ";B;CHR$(8);". ";A*B;"ITEMS."
1180 GOSUB 3300
1190 GOSUB 9360:A4=0:A5=0:A6=0:A7=0
1210 IF LEFT$(A$,1)="N"THEN1260
1220 IF LEFT$(A$,1)<>"Y"THEN1190
1230 INPUT "     ENTER `Y' IF PROGRESSIVE ",A$
1240 IF A$="Y"THEN1180
1250 CHAIN STAT1
1260 ? TAB(20);"STATPACK END": END
3300 REM PROGRESSIVE AVERAGE CALCULATION
3310 GOSUB 9400:A4=0
3320 IF B2<1 THENRETURN
3325 GOSUB 9380:?#(Z9),"PROGRESSIVE AVERAGE FOR COLUMN # ";B2
3330 ? #(Z9),"ITEM","VALUE","TOTAL","AVERAGE"
3340 FOR I=1TOA:A4=A4+C(I,B2)
3350 B3(I)=A4/I
3360 ? #(Z9),I,C(I,B2),A4,B3(I)
3370 NEXT I:?#(Z9):?#(Z9)
3380 INPUT "   ENTER `Y' FOR CHART",A$
3390 IF A$<>"Y"THENRETURN
3400 REM PRINTOUT OF DATA
3405 GOSUB 9380
3410 ? #(Z9),TAB(15);"PROGRESIVE AVERAGE CHART"
3420 ? #(Z9):A4=9.9E-99:A5=9.9E99
3430 FOR I=1TOA:IFB3(I)>A4 THENA4=B3(I)
3440 NEXT I
3450 FOR I=1TOA:IFB3(I)<A5 THENA5=B3(I)
3460 NEXT I
3470 A7=0: IFA5<0 THENA7=1
3480 A8=A4
3490 FOR I=12TO1STEP-1
3500 ? #(Z9),INT((A8*100)+.5)/100;
3510 IF A7=1 THEN3530
3520 A6=A4/12: GOTO3540
3530 A6=(A4-A5)/12
3540 FOR J=1TOA
3550 ? #(Z9),TAB((J*2)+10);
3560 IF B3(J)>=A6*I THEN?#(Z9),"*";: GOTO3590
3570 IF I=1 THEN?#(Z9),".";:GOTO3590
3580 ? #(Z9)," ";
3590 NEXT J: ?#(Z9):A8=A8-A6: NEXTI
3600 FOR I=1TO5:?#(Z9):NEXTI:RETURN
9000 REM SUB PROGRAMS
9360 INPUT "     ENTER `Y' FOR MORE, `N' FOR NONE ",A$:RETURN
9380 ? CHR$(25);:?CHR$(25);:?CHR$(22);:?CHR$(12);:RETURN
9400 IF B=1THENB2=1:GOTO9430
9410 ? :INPUT"     COLUMN # STATISTIC REQUIRED FOR ",B2
9420 IF B2>B THEN?"TOO HIGH";:GOTO9410
9430 ? #(Z9):RETURN
9600 OPEN #10, STATFL1 FOR INPUT
9610 OPEN #20,STATFL2 FOR INPUT
9620 FIELD #10,F=6
9630 FIELD #20,A=6,B=6
9640 SET #10=1:SET#20=1:GET#20
9650 DIM C(A,B),B3(A)
9660 FOR I=1TOA:FORJ=1TOB:GET#10:C(I,J)=F:NEXTJ:NEXTI
9683 INPUT "     ENTER `Y' FOR VISUAL OF DATA",X$:IFX$<>"Y"THEN9690
9685 ? :FORI=1TOA:FORJ=1TOB:?C(I,J);:NEXTJ:?:NEXTI:?
9690 CLOSE #10:CLOSE#20:RETURN
```

```
THE GHOST

  CHAINSTAT5       ENTER OUTPUT PORT # ? 1
                            STAT5 FOR PROGRESSIVE AVERAGE

                   ENTER `Y' FOR VISUAL OF DATA? Y

  3 2 0 0 2 3 1 1 50 5 0 0 0 1 0 0 0 0 0
  3 1 0 0 2 4 1 1 100 2 0 2 0 0 0 0 0 1 0
  3 1 0 1 0 1 1 1 100 3 3 2 0 1 0 1 0 1 1
  3 1 0 1 0 2 1 1 2 1 0 2 0 0 0 1 0 1 0
  3 1 0 0 2 3 1 3 2 1 0 0 0 0 1 1 0 0 0
  3 1 1 0 0 2 1 1 50 2 0 0 0 0 1 1 0 1 0

  YOUR ARRAY IS SET AT 6 x 19. 114 ITEMS.

     COLUMN # STATISTIC REQUIRED FOR ? 6

  PROGRESSIVE AVERAGE FOR COLUMN # 6
  ITEM          VALUE         TOTAL         AVERAGE
  1             3             3             3
  2             4             7             3.5
  3             1             8             2.66666666
  4             2             10            2.5
  5             3             13            2.6
  6             2             15            2.5

    ENTER `Y' FOR CHART? Y
                PROGRESIVE AVERAGE CHART

  3.5          *
  3.21         *
  2.92       * *
  2.62       * * *
  2.33       * * * * * *
  2.04       * * * * * *
  1.75       * * * * * *
  1.46       * * * * * *
  1.17       * * * * * *
  0.87       * * * * * *
  0.58       * * * * * *
  0.29       * * * * * *

     ENTER `Y' FOR MORE, `N' FOR NONE ? N
                STATPACK END

  THE GHOST
```

RANDOM



WRITINGS

Michael James

If the idea of a book, a fairly hefty volume too, containing nothing but a million "random" numbers[1] seems absurd to you then so should the idea of your computer or pocket calculator producing random digits, e.g., 1 to 6, for your computer games. The whole idea of randomness seems to be against the act of filling a book with random numbers. The fact that one could look at the same page more than once to find out what was coming wipes out the usual element of surprise that the word random contains! It is perhaps less obvious that computer generated random numbers are just as repeatable – and hence expected. In fact, some applications demand that a sequence of random numbers is repeatable. Where does all this madness take us? In short, what is random about "random" numbers?

Randomness v. pseudo-randomness

We all have a clear idea about randomness. The flip of a coin. The fall of a dice. These events are random. Their outcome is not predictable and not repeatable (at will). If we wanted to build an electronic dice into a computer, to enable us to play games say, then a direct solution would be to take some electronic device which behaved randomly, for example, the output of a Zener diode or a saline cell. We could use the device to determine the state of a memory location in our computer and bingo! we have our random number generator. This is the way ERNIE, your friendly premium bond selecting machine works. This is a very good technique for generating random numbers that nobody can ever guess at – it is the *lack of repeatability* which is important here. However, even though fair and right for the job, machines such as ERNIE are very expensive things to construct *well* and for a lot of applications a much simpler solution will do.

Let us think about the properties we would like a computer-simulated dice to have.

1) Each digit should come up, on average, as often as any other.
2) By observing which digits have come up *already* it should not be possible to predict the next digit.

Condition one concerns the fairness of the dice and condition two concerns the independence of consecutive throws. Fairness is not a difficult condition to satisfy. It is the second condition which causes all the problems. First, we should notice that there is nothing in our two conditions which says that these digits should be produced randomly in the sense of coin tossing. I could have a list of numbers in which every number occurred about equally often and in which knowledge of any set of numbers would not help me to guess the next. These would satisfy our conditions and if I read them from the list one after the other you could not tell if I was tossing a dice or reading a list. Such numbers are called *pseudo-random* numbers because they are not produced by a random mechanism and because they are, in principle, repeatable.

The requirement of not being able to predict the next random number in the sequence, given knowledge of the rest, is a problem because, if the numbers are generated by a non-random method, i.e. they are repeatable, then there *must* exist a method of predicting them! (This is, of course, using another *copy* of the program or list which gave rise to them in the first place.) So it seems that we cannot meet the second requirement. On closer examination it is obvious that we are asking too much of our random numbers – all we need is that they are not predictable in the circumstance that we are using them in. For example, if the method of prediction is either too obscure to be deduced or too difficult to be used by a human then our random numbers are O.K. for game playing on a computer. (For most applications we usually settle for successive numbers being uncorrelated with one another).

Generating random numbers

One of the first computer (pseudo) random number generators, the mid-square method, was suggested by Von Neumann in 1951. It is easy to use but generates fairly low quality random numbers – it has a tendency to produce numbers like 00XY and XY00 periodically, but it is easy to understand:

1) Specify the number of digits to be generated – say four. 2) Choose any starting value – 5069. 3) Square the starting value – 25694761. 4) The next random number is in the middle four digits – 6947. 5) Steps 3 and 4 are repeated with the new random number.

A short BASIC program for the mid-square method is given below and the reader might have some fun experimenting with it.

Mid-square

```
10  INPUT "STARTING VALUE", A
20  L = LEN(STR$(A))
30  P = 10↑(INT(L/2))
40  Q = 10↑L
50  A = INT(A*A/P)
60  A = A – INT(A/Q)*Q
70  PRINT A
80  GO TO 50
```

The most popular type of random number generator in use today is the so-called congruential generator. It is not as easy to understand as the mid-square method but it does give high quality numbers with known properties. A typical generator is given below as a BASIC program. (This particular generator is also of historic interest as it was first used on ENIAC.)

Congruential

```
10 INPUT "STARTING VALUE", A
20 A = A*23
30 A = A – INT(A/100000001)
                      *100000001
40 U = A/100000001
50 PRINT A,U
60 GO TO 20
```

The general congruential generator works by multiplying the old random number by a constant and then expressing it modulo some other constant to get the new random number, i.e.

$$A_{n+1} = \left[A_n * K\right] \bmod P$$

Expressing a number modulo P is simply done by finding the remainder after dividing by P. In our example K = 23 and P = 100000001. A further refinement is to divide A_n by P to give a random number between 0 and 1 (U in our example). Congruential generators repeat themselves eventually but this can take a long time and depends on the choice of K, P and A_1. (Our example can generate 5,882,352 numbers before repeating.) Constructing a very good congruential generator is difficult, but our example will do for most applications.

Monte Carlo

Random numbers can be used to solve some types of mathematical problems as well as in computer game playing. For example, suppose we are about to design a garage and we want to decide how many petrol pumps to install. Too many and some will stand idle and we could have saved our capital. Too few and we will lose customers as the queues get longer. Putting this another way, what we need to know is the average length of the queue for various numbers of pumps. The answer to this problem depends on the number of customers per second and the time it takes to serve them. It is not easy to get the answer by the usual mathematical methods.

A method of solving the problem is to *simulate* it using a random number generator. By writing a program in which customers arrived and were served with the right probabilities, we could obtain answers simply by running the program and keeping a count of the number of customers served and turned away.

The collection of methods based on using random numbers to solve mathematical problems is generally called the Monte Carlo method. The previous simulation example is easy to understand and the role of the random numbers is obvious. However, random numbers can be used to solve problems which seem to have nothing to do with randomness.

For example, suppose we wish to evaluate

$$\theta = \int_0^1 x^2 dx$$

In other words, find the area below the graph of x^2 in the interval 0 to 1 (Fig. 1). We could use the usual methods of numerical integration, i.e. Simpson's Rule, or even solve the problem directly by $x^2 = \frac{1}{3}x^3$. But suppose we instead generate two random numbers U_1 and U_2 which define a point in the unit square, i.e. they are both positive and less than one (see Fig. 1). If U_1 and U_2 are evenly distributed then the probability of the random point being below the curve is exactly equal to the area beneath the curve. Thus if we generate N random points, the area under the curve is estimated by the probability H/N where H is the number of points below the curve. A BASIC program to carry out this method for x^2 is given below.

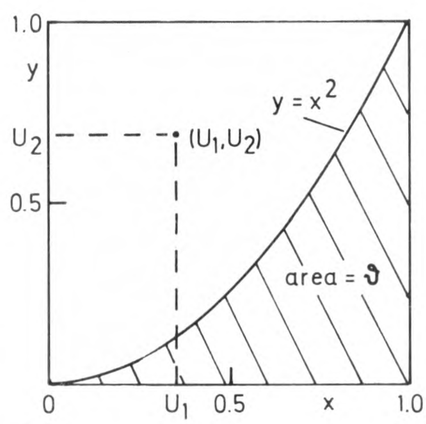


Fig. 1

Integration program

```
10  H = 0
20  N = 0
30  DEF FNA (X) = X*X
40  U1 = RND
50  U2 = RND
60  IF U2 < FNA (U1) THEN H = H + 1
70  N = N + 1
80  PRINT "AREA ESTIMATE = ",
      H/N, "N = "; N
90  GO TO 40
```

It is easy to find any one-dimensional integral over 0 - 1 by changing the function (FNA) statement. A quick look at the program shows that the method is simple when compared with other methods. However, a little experimentation will soon reveal its disadvantage – you have to do a lot of work to get a reasonable answer. For example, at N = 100 the estimate was 0.435 and even at 1000 it was only 0.361. (The correct value is 0.333). This would seem to make Monte Carlo integration of little use, but with a few improvements it is one of the best techniques we have for high-dimensional integrals. It is rarely used for one-dimensional integrations, i.e. finding areas, but it is nearly always preferred for two-dimensional cases, i.e. finding volumes.

There are other examples of turning a non-random problem into a random one and then solving by simulation but the reader is referred to the suggested reading at the end of this article for more details.

Testing random number generators

Whenever you use a random number generator you should *always* satisfy yourself that it is good enough for your purpose. This can be done either by statistical tests or, for the least exacting work, simply by examining a histogram of the output.

For game playing most random number generators are good enough. For the various Monte Carlo techniques it is advisable to conduct statistical tests before relying on the results. (Details of these tests can be found in the further reading.) I have tested a number of random number generators supplied with various versions of BASIC and found them all reasonable – none of them have been as bad as the mid-square method! One annoying feature of some BASIC random number generators is their randomisation. By starting the generator off with a new starting value, obtained from some arbitrary memory location, we lose the repeatability of pseudo-random numbers. This is excellent for game playing – otherwise you'd play the same game every time – but for Monte Carlo methods this is a nuisance. It is impossible to say how good such a randomised generator is because its properties *depend* on the starting value used.

Conclusion

Random numbers play an important part in the personal computer revolution. For game playing the random number generator supplied with BASIC (or some other high-level language) is usually good enough. Monte Carlo simulation techniques increase the usefulness of random numbers but also require better generators. A good Monte Carlo simulation is simple, effective and can be fun – after all a computer game is usually nothing more than a Monte Carlo simulation of some "real" game.

Reference

1. *A Million Random Digits with 100,000 Normal Deviates,* The Rand Corporation, 1955.

Further Reading

The Generation of Random Variates, T. G. Newman and P. L. Odell, Griffin, 1971.

The Monte Carlo Method, ed. Yu. A. Schreider, Pergamon Press, 1966.

PCW OPEN PAGE

THE AMATEUR VIEW

Mike Lord

LOCAL INTERESTS

The *Merseyside Microcomputer Group* is now flourishing to such an extent that even bigger rooms are having to be booked for their monthly meetings. Special Interest groups are being formed to cater for NASCOM, PET and Z-80 devotees, and for people interested in the uses of computers in education, while to keep everyone informed of the group's activities, MMG are now producing a group newsletter. Potential new members are invited to contact the chairman, Martin Beer, at the Computer Laboratory, University of Liverpool, P.O. Box 147, Liverpool L69 3BX, telephone 051 - 709 6022 ext. 2967.

Another university based group has now started in Oxford. Although it is called the *'St. John's College Microcomputer Society'*, membership is open to all in the Oxford area who care to get in touch with the society's secretary Rupert Steele at St. John's College, Oxford.

Despite atrocious weather, the *North Kent ACC* had a successful inaugural meeting at Biggin Hill in mid November, and have now established a regular series of meetings. Amateur computing enthusiasts living in the area should contact Barry Biddles, 3 Acer Road, Biggin Hill, Kent, telephone 71742.

Twenty people and two computers turned up at the Cross Hands, Beechwood, Newport, where, in convivial surroundings, they agreed to form the *Gwent Group* and to arrange talks and visits to places of computing interest such as the Llanwern steelworks. Pete Hesketh has all the details for anyone who cares to ring him on Shirenewton (02917) 596.

Recent meetings of the *Exeter and District ACC* have been attended by more than 50 people, and details of future meetings may be obtained from David Carne of 44 George Street, Exmouth (telephone Exmouth 74479). After some experimentation they have now settled on a format for the meetings consisting of a talk on a particular processor or piece of equipment followed by 'RAM-Time'. This is a question and answer forum at which any member is welcome to ask any question on computing and any member may answer. This seems to be an idea which could usefully be adopted by other groups.

Pete Harris of 119 Carpenter Way, Potters Bar, Herts telephone 01 - 248 8000 ext. 7065, reports that although he has had an encouraging response to the mention in this column a few months ago, he feels that there must be more readers who would like to join a 'personal' PDP/LSI11 User Group to promote the interchange of ideas and expertise on these classic machines.

A new group has been formed in the *East End of London* by Jim Turner of 63 Millais Road, London E11, and will be holding meetings on the 20th February and the 20th March in the Meeting Room of the Harrow Green Library from 7 to 10 p.m. The Library is at the Leytonstone Road end of Cathall Road in Leytonstone.

Readers living near or visiting *Hamburg* are invited to meetings of the Hamburg CC, which meets on the first Wednesday of each month. Ring Pete Bendall on (04191) 6538 and he'll put you in touch with the club.

Closer to PCW headquarters, a *South East London Group* has been formed by Roy Mitchell of 58 Kenilworth Gardens, Shooters Hill, London SE18 3JB (01 - 856 2489).

The *Newcastle Personal Computer Society* is holding meetings on the first Tuesday of each month, usually comprising a lecture, informal discussion, and the demonstration of a particular microprocessor system. For further details ring Dr. W. G. Allen on 0632 851528.

NATIONAL INTEREST

In the September issue of PCW, this column raised the question of a new standard for cassette tape recordings — on the basis that the de-facto standard: CUTS, is agonisingly slow and there are a host of later developments now available for comparison. Readers will recall that Alan Secker volunteered to compare proposals and organise a debate on the subject.

Inevitably, issues such as this take a while to resolve; however an interim report from Alan indicates that there appear to be two designs which meet the original requirements of simplicity, reliability and acceptability, and these are the 'Tarbell' system already popular in the U.S.A., and a high speed version of CUTS which uses the same frequencies but less cycles per bit, as described by Bob Cottis and Mike Blandford in the December issue of PCW. The investigation continues.

CLEAR TO SEND

If you want the world to know of your local group, SIG, or any other activity of interest to the amateur computing enthusiast, just drop a line to Mike Lord, 7 Dordells, Basildon, Essex.

SETTING UP A LOCAL GROUP

James A. M. Cunningham
Chairman, **ACC**

During the last Personal Computer World exhibition, two of the most persistent questions asked at the Amateur Computer Club stand were:

"Is there a local group for me to join?"
"How do I start one?"

This small article is an attempt to satisfy the above two questions.

The Amateur Computer Club (ACC) was set up to provide, by using a news-letter, a forum to exchange information between members. This has resulted in a national membership scattered all over Britain, but with the membership now greater than 1700 there cannot be many members who are totally isolated. Thus, provided members can meet one another there should be little difficulty in forming local groups based on a fairly compact area, so anyone who resides in an area in which a local club does not operate and wishes to belong to one should think about starting one himself/herself. However, the Amateur Computer Club by its very nature cannot assist in setting up local groups as the manpower is not available.

The first action should be to attempt to contact those of like mind in their local, easy travelling distance, area. To do this there are several methods:

(i) a small advertisement in the local paper.
(ii) write to be mentioned in the newsletter and magazines such as *Practical Electronics, Personal Computer World* etc.

When the replies start pouring in, the next job is to decide where to hold your first meeting. The size of the place required will depend on the number of people who replied. A suitable place would be a Public House, which would provide refreshments for all concerned! Although, it's surprising how many people can be accommodated in the lounge of a house for a first meeting.

If you are going to form a club, you will need officers to run it, these being a Chairman, Secretary and a Treasurer. At the beginning and especially if the number of possible members is small, all that is needed is someone who could be loosely called Secretary, to call meetings and indicate when to have them. The Chairman and Treasurer really come into their own when the Club is large enough to justify a constitution, and the Secretary is being over-burdened with work. When this occurs, the following personnel will be required: Chairman, Secretary, Treasurer, as officers; and usually three to five others as a Committee to assist the three officers.

It is important not to have too large a committee as the committee meetings can take too long and very little is decided owing to conflicting opinions. If this happens, reduce the size of the committee. If, however, the reverse happens you can always co-opt another member on to the committee to reduce the burden of work.

On choosing the meeting place, there are two main considerations to be determined. One is to try and choose some place central for the majority of those who are members, and secondly it all depends on what meeting places are available to hold meetings in. The problem of meeting places will arise quite often as the club grows.

The secretary's duties are reasonably simple but can be hard work. He is normally responsible for

i) calling committee meetings and providing agendas,
ii) taking minutes at meetings and distributing them if necessary,
iii) ensuring that the speaker gets to the general meeting on time and that any equipment requested is available,
iv) announcing the speaker and any other notices he has to hand, (sometimes the Chairman does this),
v) ensuring that the Chairman is kept up to date with any correspondence etc., likely to affect the club,
vi) co-ordinating the activities of the committee members.

The Chairman has a job which, if the secretary is up to his/her job, is fairly easy but it does require qualities which are not present in any other job. It is his job to ensure that at committee meetings the agenda is adhered to, that conflicting arguments are not allowed to get out of hand and away from the subject under discussion. It is sometimes necessary to point out that the person speaking is wandering away from the subject (usually because he tries to make things too complicated). It is also necessary for the chairman to think ahead and to attempt to foresee if the club is heading in the correct direction bearing in mind the membership who are usually of varying experience from expert (small number) to beginner (large number). There is little use having speakers who give talks which are over the heads of most of the members present.

He should also give some thought to his secretary and who he could get in his place if anything happened to him. The lack of a secretary for even one month could cause a lot of unnecessary confusion and upset. It is also his responsibility to periodically examine the treasurer's accounts with the treasurer to ensure that all is well. It also keeps the treasurer on his toes.

The treasurer's function is to control the club's finances. To this end the club has a bank account from which money can be drawn, usually on the chairman's and treasurer's signatures. He should be able to provide at each committee meeting a brief on the finances of the club with an explanation of the drawings and deposits. He should also be able to analyse the finances so that he can give warning in advance that the club is running into a cash flow crisis. This gives the committee time to take action to correct the problem.

At the Annual General Meeting (A.G.M.) he should provide a financial and an audited balance sheet for the meeting along with his recommendations for the future.

Wherever possible a small surplus should be planned every year which will allow future committees greater freedom of action. Periodically he should present his accounts to the chairman for vetting.

A constitution will be needed to provide a basis on which the running of the Club depends.

The subjects the constitution cover are:

i) aims of the club.
ii) officers and committee – election of
iii) necessary qualifications (if any) for those joining
iv) fees, both annual and if necessary per meeting to pay for the meeting room.
v) limits of action of the officers and committee
vi) rules for the AGM and EGM
vii) any others considered required.

The Annual General Meeting (AGM) is held at the end of the club year. A typical agenda would be:

1. Chairman welcomes members to the AGM. 2. Chairman reads out the minutes of the last AGM and has them accepted. 3. Points arising. 4. Chairman's report. 5. Secretary's report. 6. Treasurer's report and accounts. This has to be proposed and accepted. 7. Election of Chairman – previous chairman carries on, the Secretary and Treasurer standing down until new Chairman takes over. 8. Election of Secretary and Treasurer. 9. Election of new committee. 10. Any business to be discussed, proposed and accepted or noted on. 11. Any other business (AOB).

On the running of the general meeting there is little concrete advice one can give as most clubs run it differently.

The most common procedure seems to be:

a) Welcome the Members to the meeting. b) Make any announcements to be made. c) Obtain reports on any activities which have taken place since the last meeting. d) Announce the speaker. e) Thank the speaker after the talk and question time. f) Close the meeting.

On the subject of speakers, amateur computing is still too new in Britain for a group of speakers to have built up, thus most clubs will have to depend on their own resources.

If a speaker is available it is usual to meet him after travelling, give him dinner, and pay his travelling expenses and hotel bill if necessary.

It is hoped that the information given in this article will be of assistance to anyone thinking of starting a club in his area. It should be pointed out, however, that the information given can be only a guide, as each club seems to be unique.

PCW **O**pen **P**age Service for Amateurs
The **POP** Service

Buy, sell, exchange. Entries are **free**, limited (in future) to not more than 50 words. The **POP** Service is on p. 69.

PCW Book Review

Michael James

VERY PERSONAL OPINIONS

A. J. Aylward

D. GRIES – COMPILER CONSTRUCTION FOR DIGITAL COMPUTERS
Wiley. £ ? (Less than £7)

The most basic text on constructing compilers, interpreters or assemblers. A must for anyone who wishes to do such work.

★ ★ ★

Buy yourself a good set of data books. It may cost a bit, but it will pay dividends. Five pounds for the TEXAS TTL 'bible', another £3 for the Intel data book and the master manuals for your own processor are essential.

Adam Osborne has written some very good books; on the whole they are worth buying, perhaps more so than any other books on micros.

Make up your mind what you want, a toy, a status symbol or a computer. If you want the latter, try IBM or one of the other well established firms. Failing that, build one for yourself. You will learn an awful lot that way. If you just want a toy or a status symbol, there are plenty of people who are willing to take your money off you.

Anything you get for free is worth what you paid for it. Two great fallacies: S-100 and BASIC. Neither are universal, standard or even much use.

Murphy's Law: "If a thing can go wrong, it will".
O'Toole's Law: "Murphy's an optimist".
Jenkinson's Law: "It won't work".
Sattinger's Law: "It works better if you plug it in".

Most people won't learn even by experience. Never underestimate the power of Human Stupidity. "The better a program is, the fewer variables it possesses". Write programs that you can understand, then they will have a chance of working.

"My program takes half the space of yours and runs three times as fast".

"Yes, but my program works, yours doesn't".

Remember the Dinosaurs. To them mammals were small, slow and over-specialised. Where are the dinosaurs now?

Perhaps computers *should* take over the world. After all, we haven't done too well with it.

If fifty thousand people believe a foolish thing it is still a foolish thing. (The S-100 ?)

Standards are there to help *you*. They mean that you can borrow my equipment and know for sure that it will work in your rig. It means that you don't have to keep on re-inventing the wheel, you can copy someone else's. Only make sure your standard *is* standard.

"If in doubt, leave it out".

I recommend the following books; they cover various ranges and various degrees of expertise. I own them all and would not recommend a book I do not or have not owned.

D. E. KNUTH – FUNDAMENTAL ALGORITHMS
Addison Wesley. £7.50

This book is just what its title suggests. It is a gentle introduction to the art of computer programming. Though it contains much maths, do not let this put you off, little of it is needed for the bulk of the work covered.

PETER WEGNER – PROGRAMMING LANGUAGES, Information Structures and Machine Organisation
McGraw Hill. £3.75

An easy, though by no means a layman's introduction to a variety of aspects of computing, both software and hardware. A good primer for anyone past the Adam Osborne stage.

B. W. KERNIGHAN & P. J. PLAUGER – SOFTWARE TOOLS
Addison Wesley. £7.20

A fundamental text on *good* programming. Quite apart from being a collection of *tried and tested* programs, which figure amongst the most useful I have ever come across, this book serves to illustrate the best elements of programming style. Essential for anyone who thinks he will ever want to write software.

J. J. DONOVAN – SYSTEMS PROGRAMMING
McGraw Hill. £6.55

A 'Noddy's Guide' to the writing of assemblers, linkers, loaders, macro-processors and the kernel of an operating system. This contains most of the basic theory of computing and is *very* readable. Its examples gently lead the reader into writing all the essential blocks of what has been termed 'system software'.

D. LEWIN – THEORY & DESIGN OF DIGITAL COMPUTERS & LOGICAL DESIGN OF SWITCHING CIRCUITS
Nelson. £7.50 each

The 'bible' of hardware design. A tremendously well researched book with a fantastic bibliography. Very readable.

J. B. PEATMAN – THE DESIGN OF DIGITAL SYSTEMS
McGraw Hill. £6.65

Not as readable as the Lewin books, but a mine of examples of how to do things. A source book more than anything else, but still a good tutor.

MARGARET BODEN – ARTIFICIAL INTELLIGENCE AND NATURAL MAN.
Harvester Press. £4.95

An Open University set book. Extremely readable and very engrossing. A lot of book for your money. Not directly relevant to most of personal computing at the moment, but a taste of things to come.

B. RANDALL & L. J. RUSSELL – ALGOL 60 IMPLEMENTATION
Academic Press. £5.40

A how to implement ALGOL 60 book. Contains all the algorithms and flowcharts necessary to implement ALGOL on an 8, 16 or 24 bit machine.

PCW *We don't regard personal computers as toys. We don't think the S-100 is all that foolish. We don't .... but there's room in this world for dissent. Isn't there?* **PCW.**

THE CIRCUIT INSPECTOR

MICRODC, A.D.C. Circuit Analysis Program for the Small Computer User

Mike Brinson, B.Sc., Ph.D., MInstP., CEng., MIERE

Interactive Computer Aided Design (CAD) of electronic circuits has in the last decade become an established engineering technique. The simulation of circuit performance allows designers to refine the operation of their circuits before constructing and testing a circuit on the bench. The availability of microcomputers, with a BASIC language interpreter, offers the home user the opportunity to experiment with circuit analysis programs which were previously only available to engineers who had access to a large computer.

Provided a CAD analysis program is carefully written with a modular structure, the owner of a very modest computer can get started in this area of personal computing. The minimum configuration that is needed to run a small analysis program is a system which will support a BASIC interpreter, with a floating point package, and 4k of user random access memory.

MICRODC is an interactive BASIC program which can be used as a circuit design aid. The program is capable of analysing d.c. circuit performance to determine component voltages, currents and power dissipations. Circuits for analysis can include transistors. In MICRODC, transistors are modelled using d.c. networks which represent the device circuit function.

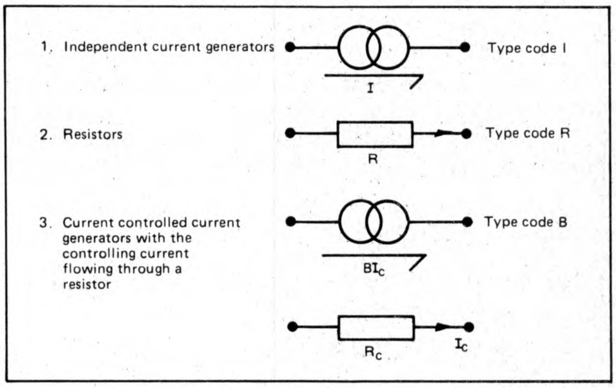
To use the program, answers to questions displayed on a VDU are entered from the VDU keyboard. These questions include *requests* for data describing the circuit components and their connection, *commands* for the d.c. analysis of circuit performance, *commands to increment* component values and *commands to modify* component values. The increment option is mainly used to observe the effects of component charges on circuit performance.

Obviously, to use MICRODC successfully a working knowledge of basic electronics is essential. To use the program, simply respond to the questions asked by the computer.

Many readers are probably asking the question: what exactly does MICRODC do?

The best way to answer this question is to consider an example. Shown in figure 1 is a simple circuit consisting of a battery and three resistors. The battery is represented in the diagram as a one volt source with an internal resistance of 0.01 ohm. For this circuit it is a simple calculation, using Ohm's law, to determine the voltage developed across each of the resistors and the current flowing in the circuit components.

However, if we were to add to the circuit series and parallel components, d.c. analysis would become more difficult and often time consuming. Given the information contained in a circuit diagram, MICRODC will automatically compute the component voltages, currents and power dissipations. The two essential pieces of information needed for these calculations are **1.** the connection of the circuit components, and **2.** the type and value of the components which form the circuit. The MK/1 version of MICRODC allows the following types of component:



The direction of assumed current flow is given by the arrows. Resistor R_C may be any circuit resistor. To specify where a component is connected in the circuit, the component interconnection points are numbered starting with zero for the earth or reference connection. The connection points are called nodes. For convenience, each type of component is also numbered but starting from one this time. Figure 2 illustrates this process using the example introduced in figure 1. In figure 2a the nodes and circuit components are shown numbered. In figure 2b since batteries are not understood by MICRODC the battery V_1 has been replaced by a current generator in parallel with R_1. Assumed current directions are indicated by the arrows. Using figure 2b as a guide the following data list is prepared.

1. Highest numbered node: 2.
2. Number of resistors: 3.
3. Number of independent current sources: 1.
4. Number of current controlled current sources: 0.
5. Specification of components and their connection:
   R,1,1,0,0.01
   R,2,1,2,1
   R,3,1,0,1
   I,1,0,1,100
   E

The end of the data list is terminated with the code E for end. Each of the resistor and independent current sources are described by the following data format:

Type code	component number	Node number from which the component current is assumed to flow	Node number to which the component current is assumed to flow	Component value

The program first displays the heading
MICRODC MK/1 Followed by
COMMAND ## ?

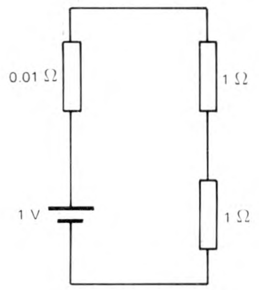


Fig. 1

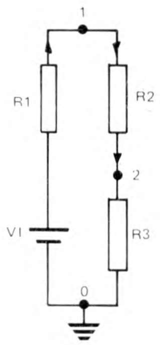


Fig. 2a

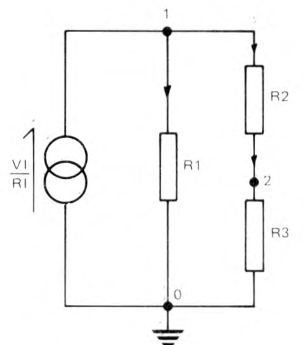


Fig. 2b

To enter circuit data, type DATA followed by a carriage return. In the following text, to help readers understand the computer output, *keyboard response are underlined.* After entering DATA the program responds with

```
NODES     =  ? 2
NO–R      =  ? 3
NO–I      =  ? 1
NO–B      =  ? 0
TYPE ? R
? 1, 1, 0, 0,01
TYPE ? R
? 2, 1, 2, 1
TYPE ? R
? 3, 1, 0, 1
Type ? I
? 1, 0, 1,100
TYPE ? E
```

The end of the data list is communicated to the program by typing the letter E followed by a carriage return. MICRODC then responds with a request for a further command.

\# # COMMAND # # ?

Responding with LIST and a carriage return will display the stored data. After the circuit data has been displayed on the VDU, MICRODC will again request a further command. A d.c. analysis of the stored circuit data is carried out by responding with DC and a carriage return. MICRODC then displays

NODE NO	NODE VOLTAGE
1	0.995024874
2	0.497512439

This output tells the user the voltage at each circuit node with respect to the earth or reference node.

On completion of each task MICRODC will request the input of a new command. The full range of options available are:

1. DATA — Enter circuit data (only used to enter data at the start of a program).
2. LIST — List stored circuit data.
3. DC — Undertake a d.c. analysis of the stored circuit data.
4. POWER — Display component voltages, currents and power dissipations.
5. MOD — Modify component values.
6. INC — Increment component values and display the node voltages for two circuit nodes.
7. FINISH — End circuit simulation.

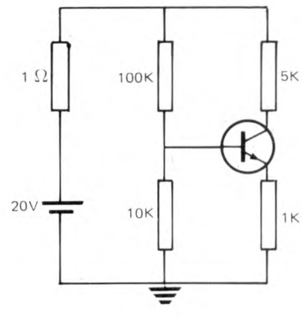


Fig. 3

In the previous example the current controlled current generator was not used. The most common use of this circuit element is to model a transistor. Shown in figure 3 is a single stage transistor amplifier. MICRODC can be used to determine the amplifier bias conditions provided we model the transistor with circuit elements the program can accept. A very basic model for the transistor is shown in figure 4. The current controlled current generator is used to represent the transistor current gain. The 200 and 30 ohm resistors are the device base and emitter resistors and the 0.6 volt battery represents V_{BE}. Figure 5 shows the amplifier circuit with the transistor model inserted and the nodes and components numbered.

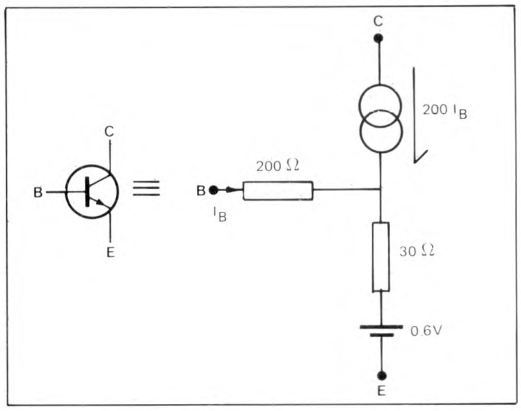


Fig. 4

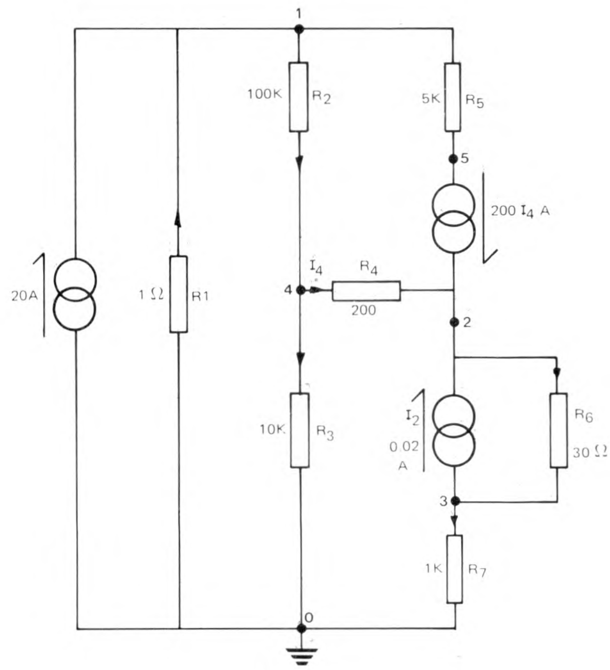


Fig. 5

The data describing the current controlled current generator is entered from the VDU keyboard with the format:

Type , code	Component , number	Node number from which the controlled current generator current is assumed to flow
node number , to which the controlled current generator current is assumed to flow	current , gain	number of resistor through which the controlling current flows

When a voltage or current is displayed as a negative number this often implies that the original assumption for the direction of current flow through a component is incorrect. If during the calculations the program finds that the current flow is in the opposite direction to that assumed in the data list, the program displays this fact as a negative number. Normally the INC command will display the node voltages for the two highest numbered nodes. However, other node voltages can be displayed by entering the code N when the INC command responds with CODE. Also if the code letter C is entered after the MOD command displays CODE the program will respond with TYPE. This facility allows the user not only to change the component value but also the position in the circuit where the component is connected. However, extra components cannot be added to the circuit. If you try to do this an array subscript error will result. Remember in BASIC an array can only be dimensioned once during program operation.

The BASIC listing for MICRODC MK/1 is given in Table 1. The program was developed on a SWTPC 6800 system using the 8k 2.0 BASIC interpreter. MICRODC will operate with other interpreters provided they are similar, with for example full floating point arithmetic, functions and string handling. A computer with either 12k or 16k random access memory is needed to run part or the complete package. To save space, dynamic arrays are used by MICRODC to store the circuit data, program lines 1050 to 1053. Hence the exact amount of memory used by MICRODC will depend on the number of analysis options included and on the size of the circuit being analysed.

To understand the operation of parts of the program requires a specialist knowledge of circuit theory and numerical mathematics. However, for those readers interested in writing their own analysis programs the following brief notes and the listing may help to unravel the coding.

At the start of the program, lines 12 to 101, a specific option is selected and the program branches to the relevant section of the code. When a DC analysis is requested, the circuit data stored in the dimensioned arrays is used to form a set of simultaneous linear equations which relate the node voltages, independent current generators and the component values. These equations are generated by the subroutine starting at line 1500. At lines 1900 and 1901 subroutines 9000 and 9300 are entered. These two subroutines solve the simultaneous equations to determine the node voltages. Finally subroutine 2000 is used to print the results of a DC analysis. Often with a large analysis task the size of the computer memory is not large enough to store the circuit data. Also the user may not need to use the full range of analysis options. Options may be removed to create space in memory for circuit data. To remove options the following program lines should be deleted.

TO REMOVE OPTION	DELETE LINES
POWER	19 and 4001 to 4490
LIST	23 and 3401 to 3451
MOD	25 and 3706 to 3749
INC	26 and 3802 to 3898

As the majority of 8k BASIC interpreters are equipped with instructions for storing BASIC programs on cassette or disk, the analysis package can be split into a number of sections corresponding to the main routines and options. One or more of the options can then be merged with the main routines by using the BASIC APPEND instruction.

Although MICRODC MK/1 is limited in its capabilities I hope that it will encourage electronics enthusiasts to experiment with circuit simulations. A second version of the program is planned for the future. MICRODC MK/2 will again be modular with extra analysis options and an extended range of components.

The author writes: At present I am mainly working on the design of high frequency circuits over the frequency range 1 – 6 GHz. The design work is undertaken using microprocessors – in particular, the M6800 SWTPC system with mini floppy discs and 20 K core.

Table 1

```
0001 LINE= 100
0004 DEF FNA(X)=Q1+X*(Q2-Q1)/10
0005  A$="NODE VOLTAGES"
0006  D$="NODE NO"
0012 PRINT "MICRODC MK/1"
0018 INPUT "£££ COMMAND £££    ",C$
0019 IF C$="POWER"THEN4001
0020 IF C$="FINISH"THENSTOP
0021 IF C$="DC"THEN1490
0022 IF C$="DATA"THEN1001
0023 IF C$="LIST"THEN3401
0025 IF C$="MOD"THEN3706
0026 IF C$="INC"THEN3802
0101 GOTO 18
1001 INPUT "NODES = ",N
1002 INPUT "NO-R  = ",N1
1003 INPUT "NO-I  = ",N2
1004 INPUT "NO-B  = ",N3
1050 DIM Y(N,N),B(N),V(N)
1051 DIM R1(1+N1),R2(1+N1),R(1+N1)
1052 DIM S1(1+N2),S2(1+N2),S(1+N2)
1053 DIM U1(1+N3),U2(1+N3),U3(1+N3),U4(1+N3)
1080 INPUT "TYPE",C$
1082 IF C$="R"THENINPUTI,R1(I),R2(I),R(I)
1083 IF C$="E"THEN18
1084 IF C$="I"THENINPUTI,S1(I),S2(I),S(I)
1085 IF C$="B"THENINPUTI,U1(I),U2(I),U3(I),U4(I)
1196 GOTO 1080
1490 GOSUB 1500
1492 GOSUB 2000
1493 GOTO 18
1500  M1=0
1510 FOR I=1TON
1512  V(I)=0
1514 NEXT I
1516 FOR I=1TON
1518  B(I)=0
1520 FOR J=1TON
1522  Y(I,J)=0
1523 NEXT J
1524 NEXT I
1532 FOR K=1TON1
1534  I=R1(K)
1536  J=R2(K)
1538  G=1/R(K)
1540 GOSUB 8500
1542 NEXT K
1552 FOR K=1TON2
1554  I=S1(K)
1556  J=S2(K)
1558  G=S(K)
1560 GOSUB 8520
1562 NEXT K
1563 IF N3=0THEN1900
1564 FOR K=1TON3
1565  I=U1(K)
1566  J=U2(K)
1567  L=R1(U4(K))
1568  M=R2(U4(K))
1569  G=U3(K)/R(U4(K))
1570 GOSUB 8530
1571 NEXT K
1900 GOSUB 9000
1901 GOSUB 9300
1902  E=0
1903 FOR J=1TON
1904  E=E+(V(J)-B(J))*(V(J)-B(J))
1905  V(J)=B(J)
1906 NEXT J
1907  M1=M1+1
1908 IF E<1E-3THENRETURN
1909 IF M1<15THEN1516
```

Cont. p66

```
1921 STOP
2000 PRINT D$,A$
2003 FOR I=1TON
2004 PRINT I,V(I)
2005 NEXT I
2006 RETURN
3401 FOR I=1TO30
3402 IF I<=N1THENPRINT"R ";I;R1(I);R2(I);R(I)
3403 IF I<=N2THENPRINT"I ";I;S1(I);S2(I);S(I)
3404 IF I<=N3THENPRINT"B ";I;U1(I);U2(I);U3(I);U4(I)
3450 NEXT I
3451 GOTO 18
3706 INPUT "CODE ",C$
3707 IF C$="R"THENINPUT"R-NO R ",I,R(I)
3708 IF C$="I"THENINPUT"I-NO I ",I,S(I)
3709 IF C$="E"THEN18
3710 IF C$="C"THEN1080
3711 IF C$="B"THENINPUT"B-NO B",I,U3(I)
3749 GOTO 3706
3802  O1=N-1
3803  O2=N
3804 INPUT "CODE ",C$
3805 IF C$="R"THEN3860
3806 IF C$="I"THEN3874
3807 IF C$="N"THENINPUT"N1 N2 ",O1,O2
3808 IF C$="E"THEN18
3809 IF C$="B"THEN3890
3851 GOTO 3804
3860 INPUT "R-NO RS RF ",Z,Q1,Q2
3862 PRINT A$
3863 PRINT "R-NO ";Z,D$;O1,D$;O2
3864 FOR X=0TO10
3865  R(Z)=FNA(X)
3866 GOSUB 1500
3867 PRINT R(Z),V(O1),V(O2)
3868 NEXT X
3869 GOTO 3804
3874 INPUT "I-NO IS IF ",Z,Q1,Q2
3876 PRINT A$
3877 PRINT "I-NO ";Z,D$;O1,D$;O2
3878 FOR X=0TO10
3879  S(Z)=FNA(X)
3880 GOSUB 1500
3881 PRINT S(Z),V(O1),V(O2)
3882 NEXT X
3883 GOTO 3804
3890 INPUT "B-NO BS BF",Z,Q1,Q2
3891 PRINT A$
3892 PRINT "B-NO ";Z,D$;O1,D$;O2
3893 FOR X=0TO10
3894  U3(Z)=FNA(X)
3895 GOSUB 1500
3896 PRINT U3(Z),V(O1),V(O2)
3897 NEXT X
3898 GOTO 3804
4001 PRINT "CODE ","VOLTAGE","CURRENT","POWER"
4003 FOR I=1TON1
4004  Z1=R1(I)
4005  Z2=R2(I)
4006 GOSUB 4450
4007 PRINT "R ";I,Z,Z/R(I),Z*Z/R(I)
4008 NEXT I
4010 FOR I=1TON2
4011  Z1=S1(I)
4012  Z2=S2(I)
4013 GOSUB 4450
4014 PRINT "I ";I,Z,S(I),Z*S(I)
4015 NEXT I
4016 IF N3=0THEN18
4017 FOR I=1TON3
4018  Z1=U1(I)
4019  Z2=U2(I)
4020 GOSUB 4450
4021  Z3=Z
4022  Z1=R1(U4(I))
4023  Z2=R2(U4(I))
4024 GOSUB 4450
4025  Z=Z/R(U4(I))
4026 PRINT "B ";I,Z3,Z*U3(I),Z3*Z*U3(I)
4027 NEXT I
4449 GOTO 18
4450 IF Z1=0THEN4454
4451 IF Z2=0THEN4456
4452  Z=V(Z1)-V(Z2)
4453 RETURN
4454  Z=-V(Z2)
4455 RETURN
4456  Z=V(Z1)
4457 RETURN
4490 GOTO 18
8500 IF I=0THEN8504
8502  Y(I,I)=Y(I,I)+G
8504 IF J=0THEN8514
8506  Y(J,J)=Y(J,J)+G
8508 IF I=0THEN8514
8510  Y(J,I)=Y(J,I)-G
8512  Y(I,J)=Y(I,J)-G
8514 RETURN
8520 IF I=0THEN8524
8522  B(I)=B(I)-G
8524 IF J=0THENRETURN
8526  B(J)=B(J)+G
8528 RETURN
8530 IF I=0THEN8535
8531 IF L=0THEN8533
8532  Y(I,L)=Y(I,L)+G
8533 IF M=0THEN8535
8534  Y(I,M)=Y(I,M)-G
8535 IF J=0THENRETURN
8536 IF M=0THEN8538
8537  Y(J,M)=Y(J,M)+G
8538 IF L=0THENRETURN
8539  Y(J,L)=Y(J,L)-G
8540 RETURN
9000  I=0
9010  I=I+1
9020  I1=I+1
9030  I2=I-1
9040 FOR L=I1TON
9050 IF I2=0THEN9090
9060 FOR J=1TOI2
9070  Y(I,L)=Y(I,L)-Y(I,J)*Y(J,L)

9080 NEXT J
9090  Y(I,L)=Y(I,L)/Y(I,I)
9100 NEXT L
9110  L=I1
9120  I3=L-1
9130 FOR K = L TO N
9140 FOR M=1TOI3
9150  Y(K,L)=Y(K,L)-Y(K,M)*Y(M,L)
9160 NEXT M
9170 NEXT K
9180 IF L = N THEN RETURN
9190 GOTO 9010
9300 FOR K=1TON
9310  K1=K-1
9320 IF K1=0THEN9360
9330 FOR I=1TOK1
9340  B(K)=B(K)-Y(K,I)*B(I)
9350 NEXT I
9360  B(K)=B(K)/Y(K,K)
9370 NEXT K
9380  K2=N-1
9390 FOR K3=1TOK2
9400  K=N-K3
9410  K4=K+1
9420 FOR J1=K4TON
9430  B(K)=B(K)-Y(K,J1)*B(J1)
9440 NEXT J1
9450 NEXT K3
9460 RETURN
```

SINCLAIRLY YOURS

W. McIvor

MPU – Calculator Interface

Winning a Sinclair Programmable Calculator in the PCW competition in issue one presented me with a problem. What do I do with two Sinclair Programmables? I bought one shortly after they were announced. My solution was to interface one of them to my MPU system.

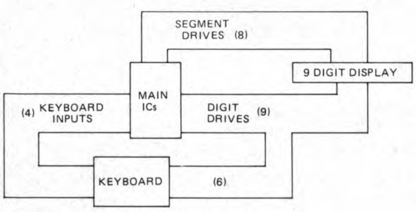


Fig. 1.

I was faced with two basic problems. Firstly, level conversion between the 9 volt calculator levels and the 5 volt MPU levels; and secondly getting the timing right.

Although based on a MPU with on-board ROM and RAM the Sinclair Programmable can be regarded as an ordinary calculator with segment and digit lines and keyboard input lines as shown in Fig. 1. Input to the calculator consists of taking one of the keyboard input lines to 0 volts when a selected digit line goes low. This is usually achieved by pressing a key which simply connects one of the digit lines to one of the keyboard input lines. Output consists of reading the segment lines when the selected digit line goes low. The calculator does this by using a multiplexed display.

These input and output methods give the block diagram of the interface shown in Fig. 2. The block 'data detect' is used to detect when the display is turned on. The Sinclair Programmable blanks the display while it is performing an operation and so when the display turns on we know that the calculator is ready to receive another instruction.

Digit Select

Figure 3 shows the circuit used to select the required digit line for input and output operations. CMOS is used here for two reasons. Firstly to avoid loading of the calculator circuit, and secondly to avoid the need to convert all 9 digit lines to 5 volt TTL levels.

A 4051 is used to select digit lines 0 to 7. The 4051 is an analogue multiplexer but it is cheaper and just as effective as the digital equivalent the

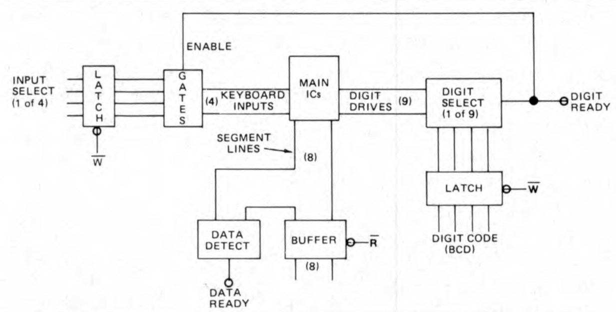


Fig. 2.

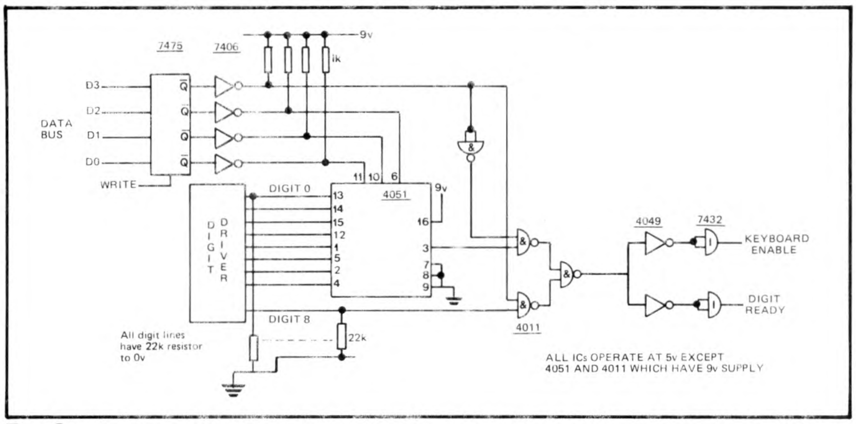


Fig. 3.

4512. Digit line 8 is selected using a 4011 quad 2 input NAND gate which selects either the output of the 4051 or digit line 8 depending on the most significant bit of the 4 bit select code.

The output of the select system is converted to 5 volt levels by using a 4049 buffer. The buffers in a 4049 when operating from a 5 volt supply give logic '1' outputs of 5 volts but will accept logic '1' inputs of 5 volts to 15 volts. The drive capability of the buffer is increased to standard TTL drive by further buffering using a gate from a 7432 quad 2 input OR gate. Separate signals are used for enabling the keyboard inputs and sending the 'digit ready' signal to the MPU, but this is simply to use up spare gates that are available.

The digit select code is written into a 7475 quad latch by the MPU and converted to 9 volt logic levels

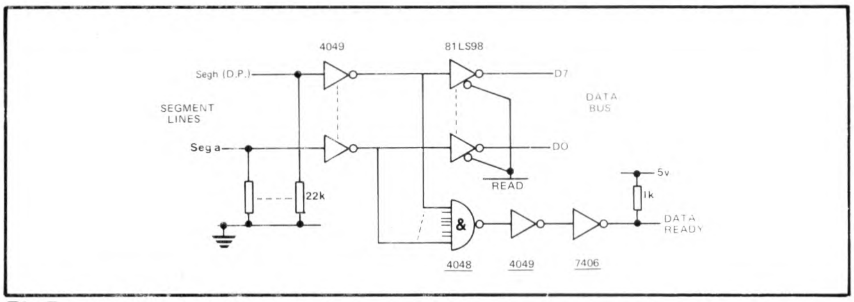


Fig 5.

Level conversion is achieved by using a 7426 which contains four, 2-input NAND gates with high voltage open collector outputs.

Segment Buffer

The segment buffer and data detect circuits are shown in Figure 5. The 4049 buffers provide level conversion to 5 volt levels. The 81LS98 provides tri-state control for connection to the MPU data bus and also reinverts

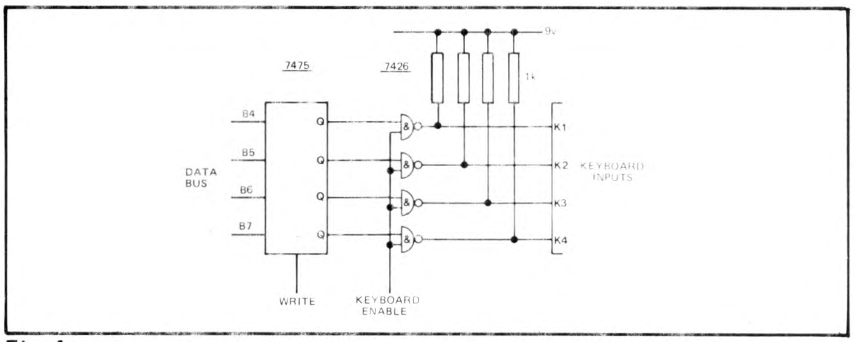


Fig. 4.

by using 7406 high voltage open collector buffers.

Keyboard Input Select

Figure 4 is the keyboard input select circuit. A logic '1' is written into the position in the latch corresponding to the input line required. When the selected digit period occurs the outputs of the latch are inverted and presented to the keyboard inputs by means of four NAND gates. Thus the bit containing a logic '1' presents a 'O' to the corresponding keyboard input during the selected digit period. Outside this period the digit select signal is at logic 'O' so the outputs of the NAND gates present logic '1' levels to the input lines.

the segment data. When the display is blanked all inputs to the 4068 are logic '1' so the data ready signal is at logic 'O'. Whenever any segment is turned on at least one input to the 4068 goes low and the data ready therefore goes high. Although this output is never continuously at logic '1', because segment data only appears for part of the digit period, it can still be used to indicate the end of an operation.

Read/Write Logic

Figure 6A shows the simple read/write logic. W and R are the negative read and write strobes from the MPU and PE is an enable signal from an address decoder. In the prototype the write signal extended beyond the time when the data on the data bus is valid because of the combined delays in the CPU circuits and the calculator interface. If this problem occurs the circuit of Figure 6B will solve it. The circuit provides a short pulse at the beginning of the write strobe.

Connections

Figure 7 shows the PCB of the Sinclair Programmable. The easiest way to make the connections to the digit and keyboard lines is to remove the keyboard, drill holes in the appropriate pads on the PCB, solder veropins into these holes and make connections to the veropins. The three digit lines not taken to the keyboard can be taken to spare pads isolated by cutting the tracks leading to them.

The segment lines are best accessed at the current limiting resistors behind the display as shown in Figure 7B. If required, the connections to the digit lines could be made at the display, and the keyboard input lines at the main IC, taking care not to bridge tracks or overheat components. If this is done the keyboard can still be used, with zero written into the keyboard input select latch, to operate the calculator manually.

Software – Input

To simulate a key entry an 8 bit word is written to the interface. The lower 4 bits are a BCD number corresponding to the digit period required. The upper 4 bits have one bit at logic '1' corresponding to the keyboard input line required. Table 1 gives the codes required for each key.

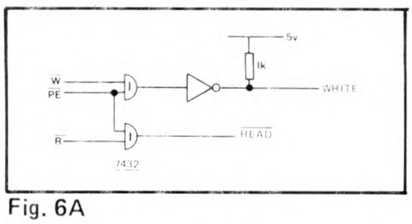


Fig. 6A

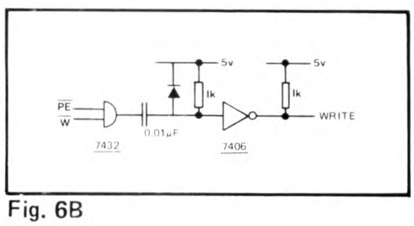


Fig. 6B

The complete input sequence is:—

1) Send 8 bit keycode.
2) Wait for calculator debounce time.
3) Send hex 00. (Key release).
4) Wait for calculator debounce time.
5) Wait for data ready to go high.
6) Send next instruction.

Hex Code	Key	Hex Code	Key
12	0	26	·/EE/–
13	1	42	▲/▲
14	2	43	C/CE
15	3	44	RUN
16	4	82	=
17	5	83	–
22	6	84	÷
23	7	85	+
24	8	86	x
25	9		

Table 1

Output

Getting the 7-segment data is more tricky. The 8 bit word has the upper 4 bits all zero and the lower 4 bits specifying the required digit. When *digit ready* goes high the segment data is ready a specific time later.

This time should be constant if a stabilised 9 volt supply is used, but as the main IC has no stable external clock it may vary for each calculator.

The output sequence therefore is:—

1) Write digit required.
2) Wait for digit ready to go low.
3) Wait for digit ready to go high.
4) Wait for segment data set up time.
5) Read segment data.
6) Read next digit.

Step 2 is in case we request data in the middle of the required digit period. The delay in step 4 was 148μs in the prototype circuit though some experimenting may be required. Also it should be noted that the *data ready* signal cannot be used for step 4 because we may be reading a blank digit which contains no segment data and does not cause data ready to go high.

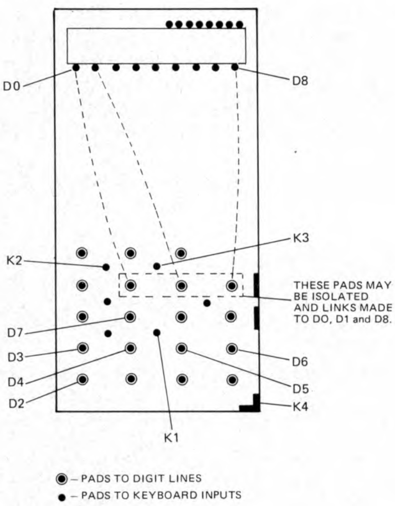


Fig 7A.

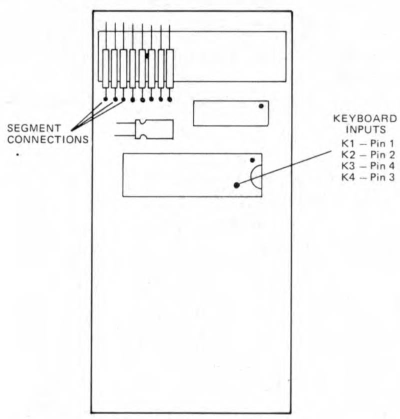


Fig. 7B

Other Software Considerations

The calculator is very slow compared with the MPU and so where possible the data ready signal should be used to interrupt the MPU. *This allows other processing to take place* while the MPU is waiting for a result.

Input and output are not compatible with each other or with any normal representation used with MPU's (i.e. ASCII, BCD, Binary) and so if the calculator is to be used, for example, in a Basic interpreter, conversion routines will be required.

A simple interpreter can be written to take a list of key codes similar to table 1 but modified to allow single and double shift functions to be specified in one byte. Special functions such as request input, display output and halt can be added. When used with a simple keyboard and display connected to the MPU, or with the calculator keyboard and display directly, the result is a system similar to the Sinclair Programmable but with the number of program steps limited only by the MPU memory. This also gives the system access to over 600 pieces of software in the form of the Program Library available for the Sinclair Programmable.

It is also possible to write a program into the calculator program memory allowing it to run a long iterative calculation while the MPU performs other tasks.

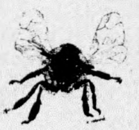
Finally, if the interface is used with any other calculator the data ready signal cannot be used. In this case a software delay is necessary to allow the operation to be completed before sending the next key entry code.

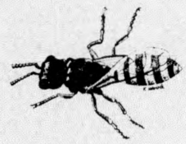
POP SERVICE

from page 59

BUZZWORDS

Peter Reynolds





Warning

Buzzwords exist to aid concise communication in an area where new ideas are emerging at an unprecedented rate, but people do occasionally use terms imprecisely, by error or intent, and sometimes the new meaning overtakes the old. That much overworked letter K, for example, stands for kilo or thousand which is often equated with 1024, the nearest round number in the binary progression which goes 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536 . . . . . So a memory with 65536 elements may be called 64k or 65k, but that is not the end of the confusion because the size of each element has not been specified. Most probably bytes of eight bits, they might equally be the bits themselves or even words of some other bit-length, such as 4, 12, 16, 24 or 32 bits. Good practice will avoid ambiguity by writing 4k x 8 bit or 4k8.

Still another ambiguity arises when old hands refer to memory as core, which is what they used to be made of, much as any domestic vacuum cleaner may be called a hoover.

So be on your guard.

E (1) Symbol for voltage, as in the equation E = IR (voltage equals current in amperes multiplied by resistance in ohms).

(2) Symbol for exponential notation, where for example 1.23 E-3 means 1.23 multiplied by 10^{-3}, or 0.00123.

e Symbol for the exponential constant, used as the base for natural logarithms.

E.13.B. A standard font for magnetic ink characters capable of interpretation by MICR reading devices.

EAROM. Electrically Alterable Read Only Memory — a form of ROM wherein the contents stored may be altered by an appropriate electrical current.

Earth. A path to earth for an electric current. It is generally necessary for a computer, in common with other electrical and electronic apparatus, to carry a connection to earth. With powerful equipment it is important that this connection be made with care in order to avoid interference with, or pickup of random signals caused, by other apparatus. (Compare the noise which may be heard on a car radio if the various parts of the motor car are not effectively suppressed and earthed by bonding to one another.) Synonymous with *ground.*

Earth Fault. An electrical fault either —
1) making a connection to earth where none should exist, or
2) in the quality of a designed earth connection, e.g. high resistance.

Earth Loop. Provision of more than one route to earth from a point in an electronic circuit. In some cases this can cause spurious signals to be picked up, like mains hum in an audio amplifier, which is most undesirable in a computer!

Earth Spike (or Spikey Earth). A transient high voltage appearing on an earth connection, normally from some external source.

Earthy (end). The conductor or connection in a set whose potential is closest to zero or that of earth, even if it is not directly connected to earth.

EBCDIC. Extended Binary Coded Decimal Interchange Code. A code, something like ASCII, used for data transmission.

Echo Check. A system of checking the accuracy of data transmission by causing the apparatus at the receiving end to send it back to the transmitter. The echo is compared with the data originally sent, and if they do not coincide some other procedure is brought into use, such as automatic re-transmission of the suspect data or an alarm signal to the operator.

ECL. *Emitter-Coupled Logic.*

ECMA 'B'. The font of natural typeface, readable by both man and machine, adopted by the European Computer Manufacturers' Association. The same as OCR-B.

Edge Card. A circuit board (or card) with contact strips along one edge, designed to mate with an edge connector.

Edge Connector. An electrical socket, slot-shaped, whereby a circuit card may be attached to a mother-board or chassis.

Edge Cutter/Trimmer. A device for removing the sprocketed margin from continuous stationery.

Edge Punched. Edge punched cards may be similar in size to conventional punch cards or a little smaller. Data is punched along the bottom edge of the card in paper tape code. This leaves the greater part of the card to be written upon freely.

Edge-punched cards are normally provided in fanfold pack joined to each other by the short edge.

Edit. To prepare data in suitable format and remove obvious errors or irrelevancies before input to an EDP system.

Editing Run. In batch processing the editing program will check the data for ostensible validity (e.g. test that dates and numbers fall within the expected ranges, compare totals with separately entered batch or hash totals and prove check digits) and identify any errors for correction and re-submission.

Editor. Computer software to make it easy to review and alter a file or program interactively. For example one editing command might locate and display the first occurrence of a given string of characters: a second command might delete or change those characters wherever they occur.

EDP. Electronic Data Processing — generally synonymous with computing.

EDS. Exchangeable Disc Store.

EHT. Extremely High Tension — a voltage likely to give a severe shock even to a person not directly connected to earth.

Electronic. Pertaining to the flow of electricity through semiconductors, valves and filters, by contrast with the free flow of current through simple conductors. The essence of computer technology is the selective use and combination of electronic apparatus whereby current can be allowed to flow or can be halted by electronic switches working at very high speed.

Emitter-Coupled Logic. Form of connecting transistors in computer circuitry (integrated or discrete). generally allowing faster and better operation.

Electronic Data Processing. A synonym for computing originally adopted to distinguish the activity from *automatic* data processing, using mechanic rather than electronic equipment. The acronym *EDP* is still popular because it is short.

Electrode. One of a set of two or more points in a device between which an electric current may flow. For example electrodes are found in batteries, in electroplating or may be applied to the human body for therapy or the measurement of voltages or skin-resistance.

Electrolyte. A liquid designed to conduct electric currents, as in a car battery.

Electrolytic. Using an *electrolyte* as for example in electrolytic condenser, a form of capacitor in which one plate is a metal surface and the other plate is electrolytic liquid, which deposits a very thin layer of insulation or dielectric on the metal surface.

Electro-Mechanical (device). Using electrical signals to trigger physical movements, for instance in an electric typewriter where touching a key closes a switch which makes the chosen letter hit the paper.

Electro-Sensitive (paper). Printer paper with a thin coating of conductive material, such as aluminium. Print becomes visible through darkening where a matrix-type print head allows electric current to flow on to the conductive surface.

Emitter. One of the three elements in an ordinary transistor, the other two being collector and base.

Emulate. To copy the performance of another, less powerful computer. Emulation, achieved by special hardware control, is similar in effect to the *simulation* attained by software. The facility may be useful when a computer is replaced by a new machine before re-writing of the old programs for conversion (almost invariably desirable for commercial routines) has been completed.

Enable. To switch a computer device or facility so that it can operate; the opposite of *disable.*

Encode. To apply a code to computer data or instructions, for example, to change values expressed in decimal figures to an experession in *excess-3 code,* or from assembler mnemonics to hexadecimal values in machine language.

Encoder. A device which produces machine-readable output, for example, paper tape, either from manual keyboard depressions or from data already recorded in some other code.

Encrypt. To make data unintelligible to those not entitled to read it by an ordered arrangement of transpositions etc., that can be restored to clarity by a device suitably programmed but difficult to manage otherwise.

End Mark/Word. Coded signal used to identify the finish of some piece of data in a variable length store.

Entry Point. A particular instruction in a program sequence at which the work may be taken up. This need not always be at the beginning and a program can have more than one entry point. There is a parallel in the programs of some washing machines which allow the user to start at an intermediate 'rinse' or 'spin dry' operation without going through the whole wash cycle.

Environment. In computing context this is more likely to refer to the mode of operation — e.g. 'in a timesharing environment' than to physical conditions of temperature, humidity etc. But either kind of environment may affect operational efficiency.

EOF. End of File.

EOT. End of Transmission: a term from Telex usage.

EPROM. Electrically Programmable Read Only Memory. The bit content of each location may be changed from 0 to 1 by a current pulse strong enough to break a fusable link.

Equivalence Element. A circuit which produces a signal if, and only if, two items of inputs are identical.

Erase. To empty an area of store of all information, leaving it ready to receive new data.

Erase code. The code for "erase" which in paper tape practice comprises a hole punched in every position. The advantage of this convention is that it can be superimposed on any other punching.

Erase Head. In a domestic tape recorder, the erase head is the device which cleans the tape of earlier signals immediately before new matter is recorded. In a computer storage device based on magnetisation of ferro-oxide surfaces (for example, tape, card or disc, but not core) the erase head operates immediately before the write head to perform a precisely similar function.

Ergonomics. The study of workers and their environment; adapting machines to the convenience of operators, with the general aim of maximum efficiency. For example, adding a numeric keypad to a standard keyboard.

Errata Slip. A list of last-minute corrections frequently found with suppliers' instruction manuals for hardware and software alike. Always mark up these corrections before following the instructions!

Error. Deviation from true value. See also *absolute error, balanced error* and *biased error. Syntax error* however means only that the rules of a programming language have been broken.

Error Code. An error message displayed by a computer in the form of a number whose significance the operator must look up in a book.

Error Detection Routine. A routine designed to detect whether or not any error has occurred in processing or operating. Detects but does not necessarily locate errors.

Error Message. A computer message, generated by its operating system or other software, to advise the operator when a fault condition is detected and (generally) to indicate how to locate and correct it.

Escape. A keyboard character (generally non-printable) which, like *shift* or *control*, translates the following character to a different character set.

Etching. The process that produces a printed circuit board from a suitably masked sheet of copper laminate.

Euro-Card. A dimensional standard (one of several choices) for a circuit board used in small processors.

Even Parity. The convention for checking data after transmission which expects an even number of ones in each group of bits transmitted.

Excess-3 Code. A system of binary numbering in which each binary equivalent is three greater than it normally would be. For example:—

Decimal Digit	*Excess-3 Code*
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

When numbers expressed in the excess-3 code are added together, "carry" digits arise at the same times as when the decimal equivalents are added; for this reason the excess-3 code is favoured in *binary coded decimal.* Another feature of excess-3 code numbers is that the code can express three negative values, viz -1, 0010; -2,0001; -3, 0000. This can be of value in certain "compare" operations.

Exchange/Sort. A system of sorting in which the key digits of two blocks of data are compared, and if they are not initially in proper sequence they are returned to store and their original locations transposed.

Exclusive "OR" Operator. *A logical operator (Boolean algebra)* which has the property that, if P and Q are two statements, the statement P*Q (where the asterisk is the exclusive OR operator) is true if either P or Q, but not both, is true and is false if P and Q are both false or both true, according to the following table. (Figure 1 indicates a binary digit or truth):

P	Q	P*Q
0	0	0 (even)
0	1	1 (odd)
1	0	1 (odd)
1	1	0 (even)

Note that the exclusive OR is the same as the *inclusive* OR except that with both statements true there is no output; that is, P*Q is true if either P or Q is true but not if both are true.

Execute. Decode a machine instruction to effect the required computer operation.

Executive (program). Generally synonymous with *operating system.*

Execution Time. The elapsed time taken by a computer to perform an instruction, such as add.

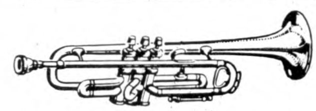
Exit. The last obeyed instruction of a routine, or the address of this instruction.

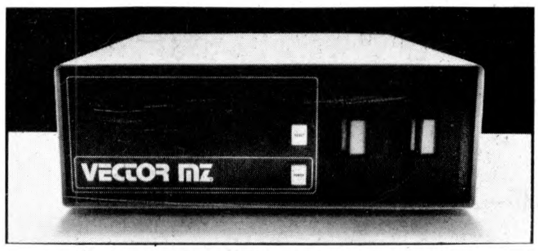
Exponent. The power to which a number is raised. Thus in 10^6 (meaning one million) the exponent is 6.

Expression. A mathematical quantity of several elements, e.g. 'SQR(A/B)', rather than a single element such as 'B' or '17'.

Extended BASIC. A version of the original Dartmouth BASIC programming language which has been enhanced by adding extra commands or facilities, e.g. to perform matrix arithmetic or to evaluate the trigonometric functions.

Extract. To take out part only of some data held in a storage area; for example, the area might hold 15 digits and the extract be confined to, say, the 8th, 9th and 10th digits.





Classified Ads

LP ENTERPRISES

Room PW
313 Kingston Road, Ilford,
Essex, IG1 1PJ, England

From the representatives in Europe . . . for America's leading Micro-computer magazines and books, for the hobbyist, educationist and professional alike, we bring you a little light browsing!

Reading maketh a full man . . . . Francis Bacon (1561 – 1626)

Introduction to Microcomputers:

Volume 0: The Beginners Book	£ 5.95
Volume 1: Basic Concepts	£ 5.95
Volume 2: Some Real Microprocessors (without binder)	£18.95
Volume 2: Some Real Microprocessors (with binder)	£24.70
Volume 3: Some Real Support Devices (without binder)	£11.95
Volume 3: Some Real Support Devices (with binder)	£17.70
6 Updating supplements for Vol. 1 (for 1 year)	£18.95
6 Updating supplements for Vol. 3 (for 1 year)	£18.95
6 Updating supplements for Vol. 2 / 6 Updating supplements for Vol. 3 (for 1 year)	£30.00
Binder (specify for Vol. 2 or Vol. 3)	£ 5.75
1 Updating supplement for Vol. 2	£ 4.00
1 Updating supplement for Vol. 3	£ 4.00

6800 Programming for Logic Design	£ 5.95
8080 Programming for Logic Design	£ 5.95
Z80 Programming for Logic Design	£ 5.95

BASIC Computer Games	£ 5.50
What To Do After You Hit Return	£ 7.00
8080 Galaxy Game	£ 7.95
The Colossal Computer Cartoon Book	£ 3.95
Computer Rage (a board game)	£ 6.95
Artist and Computer	£ 3.95
Games with a Pocket Calculator	£ 1.75
Games, Tricks and Puzzles for a Hand Calculator	£ 2.49

Z80 Instruction Handbook	£ 4.00
8080 Programmers Pocket Guide	£ 2.25
8080 Hex Code Card	£ 2.25
8080 Octal Code Card	£ 2.25

Dr. Dobb Journal Vol. 1	£10.00
Best of BYTE	£ 8.95
Scelbi BYTE Primer	£ 9.95
Best of Creative Computing: Vol. 1	£ 6.95
Best of Creative Computing: Vol. 2	£ 6.95
Best of MICRO	£ 6.95

8080A/8085 Assembly Language Programming	£ 6.95
6800 Assembler Language Programming	£ 6.95
8080 Software Gourmet Guide and Cookbook	£ 7.95
6800 Software Gourmet Guide and Cookbook	£ 7.95
Microcomputer Programming: 6502/6800	T.B.A.

MAGAZINES: Subscriptions

Subscriptions start within 3 weeks.	**U.K. Price**	**Overseas Price**
Personal Computing (Twelve Issues Yearly)	£16.00	£17.00
Interface Age (Twelve Issues Yearly)	£20.00	£20.50
Dr. Dobbs Journal (Ten Issues Yearly)	£13.00	£13.50
Computer Music Journal (Four Issues Yearly)	£ 8.50	£ 9.00
Peoples Computers (Six Issues Yearly)	£ 8.00	£ 8.50
BYTE (Twelve Issues Yearly)	£21.00	£21.50
Creative Computing (Twelve Issues Yearly)	£16.00	£16.50
Calculators and Computers (Seven Issues Yearly)	£10.00	£10.50
Kilobaud (Twelve Issues Yearly)	£20.00	£21.00
73 (Twelve Issues Yearly)	£20.00	£21.00
MICRO-6502 Journal (Six Issues Yearly)	£ 7.50	£ 8.00

First Book of Kim	£ 6.50
Microprocessors from Chips to Systems	£ 7.95
Microprocessor Interfacing Techniques	£ 7.95
Z80 Microcomputer Handbook	£ 7.50
T.V. Typewriter Cookbook	£ 7.50
T.T.L. Cookbook	£ 7.50
CMOS Cookbook	£ 7.95
IC OP-AMP Cookbook	£ 9.50
RTL Cookbook	£ 4.25

Some Common BASIC Programs	£ 5.95
Computer Programs that work (in BASIC)	£ 2.55

Introduction to Personal and Business Computing	£ 4.95
Getting Involved with Your Own Computer	£ 4.75
Your Home Computer	£ 7.95
How to Profit from your Personal Computer	£ 5.50
Reference Book of Personal + Home Computing	£ 4.95
Hobby Computers are Here	£ 3.95
New Hobby Computers	£ 3.95
Understanding Microcomputers and Small Computer Systems	£ 7.95

Instant BASIC	£ 7.50
My Computer Likes Me . . . When I speak in BASIC	£ 2.75
BASIC, BASIC	£ 6.50
Advanced BASIC	£ 6.00
Introduction to PASCAL	T.B.A.

Accounts Payable & Account Receivable	£10.95
Payroll with Cost Accounting	£10.95
General Ledger	£10.95

BASIC Software Library:	
Vol 1.: Business and Games Programs	£17.50
Vol 2: Maths, Engineering, Statistical Programs	£17.50
Vol 3: Advanced Business Programs	£26.95
Vol 4: General Purpose Programs:	£ 7.95
Vol 5: Experimenting Programs	£ 7.95
Vol 6: Miniature Business System	£32.50
Vol 7: Chess/Medbil/Wd proc Programs	£26.95

8080 Standard Monitor	£ 9.95
8080 Standard Editor	£ 9.95
8080 Standard Assembler	£15.95
Bar Code Loader for 6800, 8080, Z80, 6502 Micros	£ 1.75
Tiny Assembler for 6800 Systems	£ 5.75

MAGAZINE BACK ISSUES

Personal Computing	£ 1.75
Interface Age	£ 2.25
Dr. Dobbs Journal	£ 1.75
ROM	£ 1.75
Computer Music Journal	£ 2.50
Peoples Computers	£ 1.75

HOW TO ORDER

Please note our prices include postage and packing, but not insurance, if wanted add 12p for every £10. of **books** ordered. Make cheques, PO's etc. payable to:–

L.P. Enterprises.
CREDIT CARDS ACCEPTED
BARCLAYCARD VISA/ACCESS
DINERS CLUB/AMERICAN EXPRESS

Phone: 01-553 1001 for Credit Card orders (24–hr service)

Send to address above
Indicate Payment Method:

All Orders must be Prepaid
Total Enclosed £ . . . .

. . . . . . My Cheque, P.O., I.M.O. is enclosed in **Sterling** on **U.K. Bank**

. . . . . . Charge to Barclaycard/Visa/Access/Diners/American Express

Credit Card No . . . . Expiry Date . . . .

Name . . . .

Address . . . .

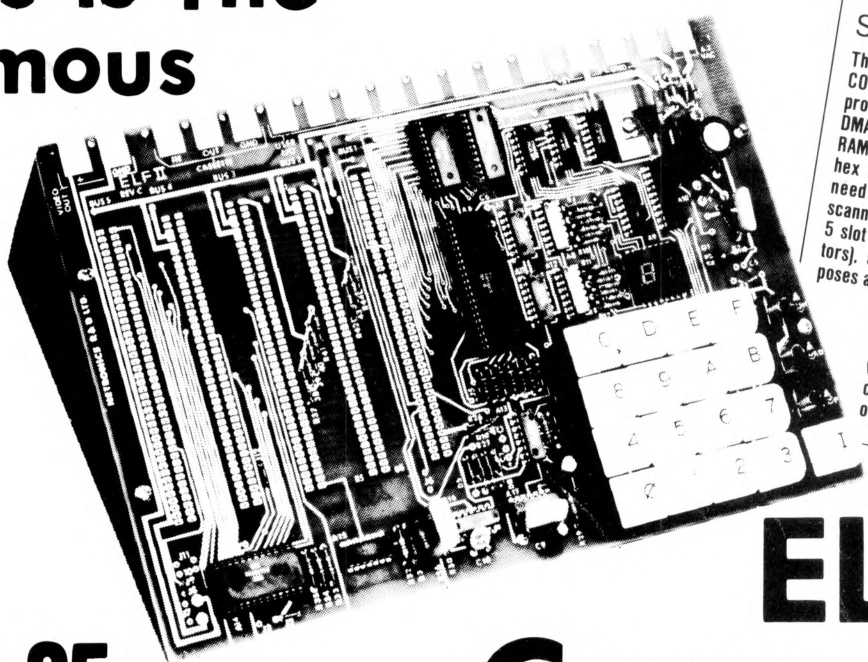
. . . . POSTCODE . . . .

Signature . . . .

All publications are published in U.S.A. and shipped air-freight by **L.P. Enterprises.** In unusual cases, processing may exceed 30 days.
*At time of going to Press, price of binders unknown. Telephone enquiries welcome.

TRADE ENQUIRIES WELCOME

This Is The Famous



SPECIFICATIONS

The £99.95 ELF II computer features an RCA COSMAC COS/MOS 1802 8-bit microprocessor addressable to 64k bytes with DMA, interrupt, 16 registers, ALU, 256 byte RAM expandable to 64K bytes, professional hex keyboard fully decoded so there's no need to waste memory with keyboard scanning circuits, built-in power regulator, 5 slot plug-in expansion bus (less connectors), stable crystal clock for timing purposes and a double-sided, plated-through pc board plus RCA 1861 video IC to display any segment of memory on a video monitor or TV screen along with all the logic and support circuitry you need to learn every one of the RCA 1802's capabilities.

ELF 11 Computer

£99.95

PLUS 8% V.A.T.

Stop reading about computers and get your hands on one! with ELF11 and our new Short Course by Tom Pittman, you can master computers in no time at all! ELF II demonstrates all 91 commands an RCA 1802 can execute and the Short Course quickly teaches you how to use each of the 1802's capabilities

ELF 11's video output lets you display an alphanumeric readout or graphics on any T V screen or video monitor and enjoy the latest T V games

But that's not all, once you've mastered computer fundamentals ELF 11 can give you POWER with add-ons that are among the most advanced found anywhere American 1EEE chapters plus hundreds of universities and major corporations have chosen the ELF 11 to introduce their students and personnel to microprocessor computing!

Learn The Skill That May Soon Be Far
More Important Than Your College Degree!

The ability to use a computer may soon be more important to your earning power than college degree Without a knowledge of computers you are always at the mercy of others when it comes to solving highly complex business, engineering, industrial and scientific problems People who understand computers can command MONEY and to get in on the action, you must learn computers Otherwise you will be left behind

ELF 11 is the F-A-S-T Way to Learn
Computer Fundamentals!

Regardless of how minimal your computer background is now, you can learn to programme a computer in almost no time at all That's because Netronics has developed a special Short Course on Microprocessor And Computer Programming in non-technical language that leads you through every one of the RCA COSMAC 1802's capabilities so you'll understand everything ELF 11 can do and how to get ELF 11 to do it!

All 91 commands that an 1802 can execute are explained to you, step-by-step The text, written for Netronics by Tom Pittman, is a tremendous advance over every other programming book in print

Keyed specifically to the ELF 11, it's loaded with "hands on" illustrations When you're finished, ELF 11 and the 1802 will no longer hold any mysteries for you

In fact, not only will you be able to use a personal computer creatively, you'll also be be able to understand computing articles in the technical press

If you work with large computers, ELF 11 and our Short Course will help you to understand what makes them tick

A DYNAMITE PACKAGE FOR JUST £99 95 plus 8% VAT!

With ELF 11, you learn to use machine language — the fundamental language of all computers Higher level languages such as FORTRAN and BASIC must be translated into machine language before a computer can understand them With ELF 11 you build a solid foundation in computers so you really know what you're doing, no matter how complicated things get

Video output also makes ELF 11 unique among computers selling for such a low price Attached to your T V set, ELF 11 becomes a fabulous home entertainments centre It's capable of providing endless hours of fun for both adults and children of all ages! ELF 11 can create graphics, alphanumeric displays and fantastic video games

Only a low cost R F Modulator is required to connect ELF 11 to your T V's aerial socket! (To order one see coupon below)

ELF 11's 5-card expansion bus (connectors not included) allows you to expand ELF 11 as your need for power grows If you're an engineer or hobbyist you can also use ELF 11 as a counter, alarm, lock, thermostat, timer or for countless other applications

ELF 11 EXPLODES INTO A GIANT!

Thanks to ongoing work by RCA and Netronics, ELF 11 add-ons are among the most advanced anywhere Plug in the GIANT BOARD and you can record and play back programmes, edit and debug programmes, communicate with remote devices and make things happen in the outside world Add Kluge Board to get ELF 11 to solve special problems such as operating a more complex alarm system or controlling a printing press Add 4k RAM Board and you can write longer programmes, store more information and solve more sophisticated problems

Expanded, ELF 11 is perfect for engineering, business, industrial, scientific and personal finance and Tax applications No other small computer anywhere near ELF 11's low price is backed by such extensive research and development programmes

The ELF-Bug Monitor is an extremely recent breakthrough that lets you debug programmes with lightning speed because the key to debugging is to know what's inside the registers of the microprocessor and, instead of single stepping through your programme, the ELF-BUG monitor, utilising break points, lets you display the entire contents of the registers on your T V screen at any point in your programme You find out immediately what's going on and can make any necessary changes Programming is further simplified by displaying 24 Bytes of RAM with full address, blinking curser, and auto scrolling A must for serious programmers! Netronics will soon be introducing the ELF 11 colour graphics and music system — more breakthroughs that ELF 11 owners will be the first to enjoy!

NOW BASIC MAKES PROGRAMMING ELF 11 EVEN EASIER!

Like all computers, ELF 11 understands only "machine language" — the language computers use to talk to each other But, to make life easier for you we have developed an ELF 11 Tiny BASIC It talks to ELF 11 in machine language for you, so you can programme ELF 11 with simple words that can be typed out on a keyboard such as PRINT RUN and LOAD

"ASK NOT WHAT YOUR COMPUTER CAN DO ... BUT WHAT IT CAN DO FOR *YOU*"

Don't be trapped into buying a dinosaur simply because you can afford it and it's big ELF 11 is more fun than "big name" computers that cost a lot more money

With ELF 11 you learn to write and run your own programmes You're never reduced to being a mere key punch operator working blindly with someone else's predeveloped software

No matter what your speciality is, owning a computer which you really know how to use is sure to make you a leader ELF 11 is the fastest way there is to get into computers Order from the coupon below!

H.L. Audio, 138 Kingsland Road, London E2 8BY

SEND TODAY

NOW AVAILABLE FOR ELF 11

- ☐ Tom Pittman's Short Course on Microprocessor & Computer Programming teaches you just about everything there is to know about ELF 11 or any RCA 1802 computer Written in non-technical language, it's a learning breakthrough for engineers and laymen alike £5 00 post paid!
- ☐ Deluxe metal cabinet with plexiglasdust cover for ELF 11 £29 95* plus £1 50 p&p R F modulator for use with T V set £3 00* post paid
- ☐ GIANT BOARD kit with cassette I/O, RS 232-C/TTY I/O, 8-bit P I/O, decoders for 14 separate I/O instructions and a system monitor/editor, £39 95 plus £1 00 p&p
- ☐ KLUGE (prototype) BOARD accepts up to 36 IC's £17 00 plus 50p p&p
- ☐ 4k static RAM kit Addressable to any 4k page to 64k £89 95 plus £1 56 p&p
- ☐ Gold plated 86-pin connectors (one required for each plug-in board) £5 70* post paid
- ☐ Professional ASC11 Keyboard kit with 128 ASC11 upper/lower case set, 96 printable characters, onboard regulator, parity, logic selection and choice of 4 handshaking signals to mate with almost any computer £64 95* post paid
- ☐ Deluxe metal cabinet for ASC11 Keyboard £19 95* plus £1 50 p&p
- ☐ ELF 11 Tiny BASIC on cassette tape Commands include SAVE, LOAD, ·, x, ÷, (), 26 variables A-Z, LET, IF/THEN, INPUT, PRINT, GO TO, GO SUB, RETURN, END, REM, CLEAR, LIST, RUN, PLOT, PEEK, POKE Comes fully documented and includes alphanumeric generator required to display alphanumeric characters directly on your T V screen without additional hardware Also plays tick-tack-toe plus a drawing game that uses ELF11's hex keyboard as a joystick 4k memory required £14 95* post paid
- ☐ Tom Pittman's Short course on Tiny BASIC for ELF11 £5 00* post paid
  Expansion Power Supply (required when adding 4k RAM) £19 95*
- ☐ ELF BUG TM Deluxe System Monitor on cassette tape Allows displaying the contents of all registers on your T V at any point in your programme Also displays 24 bytes of memory with full addresses, blinking cursor and auto scrolling A must for the serious programmer! £14 95* postpaid
  COMING SOON A-D, D-A converter, Light Pen, Controller Board, Colour Graphics & Music System and more!

*Plus VAT at 8% on price and postage

CALL OR WRITE FOR WIRED PRICES

H L AUDIO LTD , DEPT E T I
138 KINGSLAND ROAD
LONDON E2 8BY
(TEL 01-739 1582)
Sole European Distributors for Netronics
R&D Ltd U S A

YES! I want to run programmes at home and have enclosed ☐ £109 56 including postage and V A T for RCA COSMAC ELF11 kit ☐ £5 94 including postage and V A T for power supply (required) ☐ £5 95 including postage and V A T for RCA 1802 Users Manual ☐ £5 95 including postage and V A T for "Short Course On Microprocessor and Computer programming"

☐ I want mine wired and tested with power supply RCA 1802 Users Manual and Short Course included for just £164 10 including postage and V A T ☐ I am also enclosing payment (including postage and V A T) for the items checked at the left

Total enclosed ....

USE YOUR ☐ ACCESS ☐ BARCLAYCARD
Interbank
Account No ....

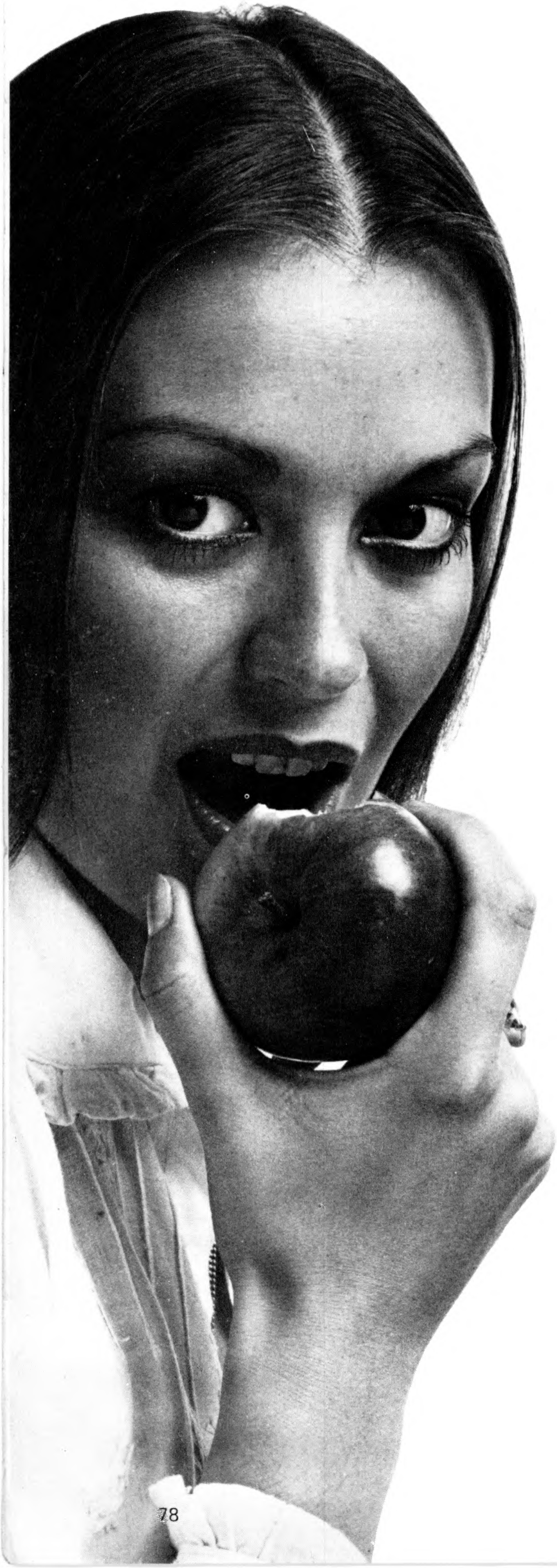
Signature .... Exp date ....

CREDIT CARD PHONE ORDERS ACCEPTED. 01-739 1582

PRINT
Name ....

Address ....

DEALER ENQUIRIES INVITED



THE SORCERER HAS ARRIVED

Introducing the personal computer you've waited for. The Exidy Sorcerer.

I didn't buy my personal computer until I found the one that had all the features I was looking for.
The Exidy Sorcerer does everything I wanted to do and a few things I never dreamed of.
It isn't magic. Exidy started with the best features of other computers, added some tricks of their own, and put it all together with more flexibility than ever before available. Presto! My reasons for waiting just disappeared.

I wanted pre-packed programs.
Software on inexpensive cassette tapes for the Sorcerer is available from Exidy and many other software makers.

I wanted user programmability
The Sorcerer's unique plug-in ROM PAC™ Cartridges contain programming languages such as Standard (Altair 8k*) BASIC, Assembler and Editor (so I can develop system software), operating systems such as DOS (so I can also use FORTRAN and COBOL) and applications packages such as Word Processor.

*Altair is a trademark of Pertec Computer Corp.

I wanted Graphics, and the Sorcerer is super. Its 256 character set — more than any other personal computer — includes 128 graphic symbols that I can define.

I wanted high resolution video.
With 122,880 points in a 512 x 240 format, I get the most detailed illustrations.

I wanted to display more information.
The Sorcerer displays 1920 characters in 30 lines of 64 characters — equal to a double-spaced typed page.

I wanted a full, professional keyboard.
The Sorcerer's 79-key data processing keyboard provides designated graphics, the complete ASCII character set in upper and lower case, and a 16-key numeric pad.

I wanted memory. The 12k of ROM holds a Power-On Monitor and Standard BASIC; 32k of RAM is supplied on board.

I wanted expandability. Serial and parallel I/Os are built in, and the optional 6-slot S-100 expansion unit lets my system grow.

I wanted a computer that's easy enough for children to use. I just connect my Sorcerer to a video display and a cassette tape recorder, and if I have any questions the easy-to-understand Operation and BASIC Programming manuals have the answers.

I wanted to buy from an experienced Manufacturer. In five years Exidy has become the third largest producer of microprocessor-based video arcade games.

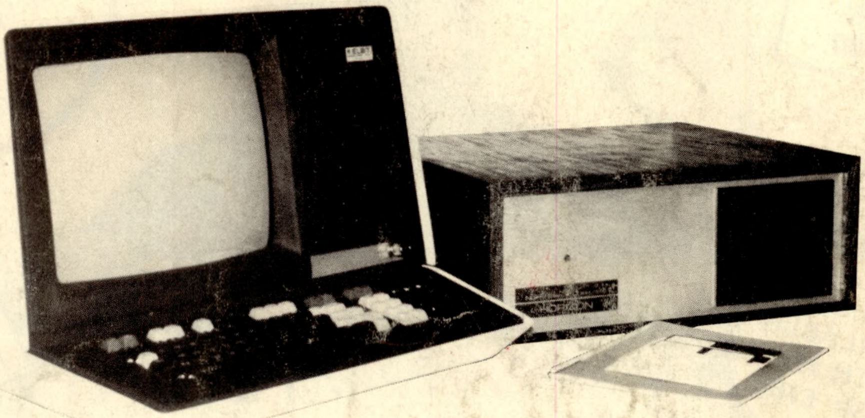
I wanted to spend less than £1,000.
(This is where COMP. does a little magic). My Sorcerer cost me £950!.
Now, what are **you** waiting for?
Call COMP. on 01-441 2922 or write to

COMP COMPUTER COMPONENTS
14 Station Road,
New Barnet, Herts. EN5 1QW.

(Price shown ex. VAT)

DEALER ENQUIRIES INVITED, A LIMITED NUMBER OF DEALERSHIPS STILL AVAILABLE.

HORIZON



THE PROFESSIONAL COMPUTER

For Business, Scientific and Educational Uses.

PROFESSIONAL HARDWARE

Use of the North Star Horizon for a short period will enable you to appreciate the professionalism in the product. There's a solid well-built chassis, a good power supply, a quiet fan and an attractive wooden case. There's a Z80A processor running at 4MHz with the 250ns static RAM boards.

There are dual integral Shugart minifloppy drives (capacity of about 360 KB on line, with an option for a further two drives), enabling easy and quick handling and copying of programs and data files.

And of course, there's the 12 slot S-100 bus which enables you to plug in many types of peripheral boards, including a hardware floating point board for increased "number crunching" performance.

PROFESSIONAL SOFTWARE

North Star Computers built their professional reputation around their powerful, but simple, Disc Operating System and Disc Extended BASIC Interpreter.

The latter contains, in addition to the usual BASIC commands, random and sequential access disc files, strings, string operators, multiple dimensioned arrays, formatted output, machine language CALL, memory EXAMine and FILL, line editor, program chaining and more.

The CP/M operating system is also available as an option and provides access to a Macro Assembler, C BASIC Compiler and FORTRAN-80 and COBOL-80 Compilers. A standard UCSD PASCAL has now been implemented.

TYPICAL APPLICATIONS SOFTWARE

- Financial
- Mathematical
- Statistical
- Educational
- Games
- Sales Ledger
- Purchase Ledger
- Stock Control
- Payroll
- General Ledger
- Estate Agents Package
- Incomplete Records
- Employment Agents

HORIZON with dual drives, 24K RAM and standard serial port – £1823. Extra 8K 250ns static memory – £155. Extra serial port – £45. Parallel port – £45.

COMPLETE HORIZON BUSINESS SYSTEM (hardware) with 24K RAM, dual mini-floppy drives, VDU and 30cps printer – £3616; 32K RAM, dual mini-floppy drives, VDU and 150cps printer £4658.

Prices exclude VAT and carriage.

Dealer, OEM and Educational Discounts available.

EQUINOX
COMPUTER SYSTEMS LTD.
32-35 FEATHERSTONE STREET
LONDON EC1Y 8QX
01-253 3781/9837